INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Stratégie perceptive d'un environnement statique dans un contexte de vision active

Eric Marchand , François Chaumette , Eric Rutten

N° 2092

octobre 1993

PROGRAMME 4

Robotique,
image
et vision

INRIA
RENNES

Stratégie perceptive d'un environnement statique dans un contexte de vision active

Eric Marchand *, François Chaumette **, Eric Rutten ***

Programme 4 — Robotique, image et vision
Projet Temis

Rapport de recherche n ° 2092 — octobre 1993 — 42 pages

Résumé : Nous étudions le problème de la reconstruction d'une scène tridimensionnelle composée de primitives paramétrables (ici, des cylindres et des droites) dans un contexte de vision active.

Après avoir présenté une méthode de reconstruction de primitives 3-D basée sur la vision active et l'asservissement visuel, on présente une méthode permettant l'enchaînement des reconstructions axée sur les informations visuelles acquises dans l'image ainsi que sur les informations obtenues lors d'une reconstruction sommaire des élements de la scène. Une ébauche du problème de l'exploration de la scène est donnée. La solution proposée, avec comme objectif d'assurer la reconstruction complète de la scène, repose sur un calcul de la zone de l'espace déjà observée par la caméra.

Enfin, une méthode de mise en œuvre d'un tel système à partir d'un réseau d'automates est présentée et son implémentation à l'aide du langage SIGNAL est réalisée. Ce réseau d'automates est capable de gérer et d'enchaîner les différentes phases de la reconstruction globale d'une scène complexe : sélection des primitives à reconstruire, reconstruction précise de ces primitives, reconstruction sommaire des autres primitives et acquisition d'informations complémentaires sur la scène.

(Abstract: pto)

*IRISA e-mail marchand@irisa.irisa.fr
**IRISA / INRIA e-mail chaumett@irisa.irisa.fr
***IRISA / INRIA projet EP-ATR e-mail rutten@irisa.irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 38 38 32

Perceptive strategy of a static environment in an active vision context

Abstract: In this report, we study the problem of 3-D structure estimation of a set of geometrical primitives in an active vision context.

The first part of this report provides a survey of the state of the art in vision guided space perception and object reconstruction. We focus on a method based on computing a structure from known motion, using visual servoing.

We then present a method for connecting up many estimations. It is centered on visual 2-D information and on a rough estimation of the structure of the others primitives. The first step for a whole exploration of the scene is given. The suggested solution is based on the Delaunay triangulation and its properties. It allows us to determine the observed space and to ensure that the whole scene has been estimated.

This method has been implemented with an automata network using the real time SIGNAL language. These automata are able to connect up the different stages of the reconstruction process (selection, focusing, optimal estimation of the selected primtive and simultaneously, rough estimation of the other ones) and to provide a robust estimation of the spatial organisation of the scene. Simulation results are given.

Table des matières

1 Introduction

Un système automatique destiné à effectuer des tâches répétitives (comme les robots manipulateurs de l'industrie) n'a pas besoin de percevoir son environnement pour fonctionner. En revanche, si ce système est destiné à des tâches d'inspection ou de cartographie, il doit le percevoir et construire une représentation correcte de celui-ci.

Une perception correcte de l'environnement est un élément clef de l'indépendance que l'on souhaite donner aux robots. Les systèmes de vision dont ils sont dotés sont en grande majorité des systèmes fournissant des informations bidimensionnelles et capables de résoudre un seul type de problèmes définis à l'avance. La reconstruction 3-D pose plus de difficultés et l'apparition de robots dits de seconde génération rend ce type de perception indispensable.

Une vision globale de la scène à partir d'un certain nombre d'images (couple stéréoscopique ou autres) ne suffit pas pour obtenir une cartographie précise du lieu étudié. Pour obtenir cette précision, le système de vision doit être capable de repérer un objet, de se focaliser dessus et dans certains cas (droites, cylindres, sphères,...) d'obtenir les paramètres qui caractérisent son équation. Dans cette optique, les travaux décrits dans [14] ont montré qu'il était possible de reconstruire des primitives géométriques 3-D paramétrables (droite, segment, cylindre, sphère,...) par vision active, ce qui permet de générer automatiquement le mouvement de la caméra nécessaire à une bonne reconstruction.

Cependant, cette approche ne permet de reconstruire qu'une primitive à la fois, le capteur devient donc une ressource "rare". Comme chez les vertébrés où le cerveau distribue les ressources rares de la fovéa à différentes tâches pouvant présenter des requêtes conflictuelles, il faut concevoir un séquenceur donnant alternativement à une tâche le controle du sytème de vision [35]. Ce séquenceur peut souvent être réduit à un simple automate (c'est souvent le cas dans les systèmes réactifs). Cependant il ne propose alors pas les capacités multitâches nécessaire pour gérer plusieurs reconstructions en parallèle . Il semble donc nécessaire de se pencher sur la conception d'un réseau d'automates capable de gérer l'ensemble du processus de reconstruction d'une scène complexe.

Ces automates devront sélectionner et gérer les actions à effectuer en fonction de plusieurs paramètres :

- les événements 2D perçus dans l'image ;
- les hypothèses *a priori* sur le contenu de la scène régies en général par l'application visée. Par exemple, dans ce rapport, nous supposerons que la scène est constituée de cylindres et de droites, notre application étant de localiser précisément les différents éléments d'une zone similaire à un environnement de type nucléaire ;
- les connaissances acquises et répertoriées au fur et à mesure de la reconstruction.

Le plan de ce rapport est organisé de la manière suivante.

- Dans un premier temps, on présentera, outre quelques techniques de perception de l'espace, différentes méthodes qui peuvent être utilisées pour la reconstruction de scènes en insistant sur la reconstruction de primitives géométriques paramétrables à l'aide de la vision active.
- On décrira ensuite les différentes étapes nécessaires à la reconstruction d'une scène complexe et en particulier le problème de la focalisation sur chaque primitive de la scène. On proposera aussi un cadre général au problème de l'exploration de la scène.
- Une troisième partie décrira une méthode de mise en œuvre de la reconstruction à l'aide d'un réseau d'automates. Des résultats seront présentés.

2 Techniques de perception d'une scène 3-D

Un environnement 3-D ne peut être perçu d'un seul coup d'œil ou à partir d'une seule image dans le cas de la vision artificielle. Il faut donc développer des stratégies de perception qui doivent permettre une vision globale de l'environnement à partir de l'analyse des informations dont on dispose *a priori* ou qui sont perçues, puis après focalisation, reconstruire localement les objets présents dans le champ de vision.

2.1 Perception de l'espace

Le mouvement de l'œil humain étant, le plus souvent, pris comme base de référence, les travaux existants peuvent se classer en deux groupes différents : la recherche d'un chemin d'exploration "émergent" (le chemin est déduit du contenu des images observées) et la création d'un chemin d'exploration "explicite" (celui-ci est généré explicitement à partir d'un modèle connu *a priori*).

2.1.1 Mouvement des yeux

Concernant la vision de l'homme, le mouvement des yeux joue un rôle important dans la perception visuelle. Une description de ce mouvement est donnée par Norton et Stark [29]. Ils ont montré qu'une personne a une manière répétitive et caractéristique pour explorer et reconnaître une scène. La reconstruction visuelle d'une scène ne se fait pas d'un seul coup, il est rarement possible d'en reconnaître tous les éléments à la fois, il faut choisir, sélectionner et se focaliser. Ceci est dû au fait qu'une information visuelle détaillée ne peut être obtenue qu'à travers la fovea [1]. C'est donc le cerveau qui ordonne aux yeux de bouger de manière à se focaliser successivement sur chaque point d'intérêt de la scène étudiée.

Quand on regarde un objet statique, les yeux alternent entre des phases de fixation et des phases de mouvements très rapides. Ces mouvements amènent les yeux sur les nouveaux points de fixation. Il faut noter que pendant cette phase de mouvements, le cerveau ne collecte pas d'informations visuelles ; celles-ci sont collectées uniquement pendant les phases de fixation.

Trois points semblent à retenir plus particulièrement :

- la mémoire conserve les primitives importantes d'un objet afin de mieux le reconnaître par la suite ;
- les points de fixation principaux dans une scène sont les angles, les bords et les lignes droites ;
- la phase d'exploration d'une scène est différente de la phase de reconnaissance. Quand on regarde une scène pour la première fois, le cerveau retient le chemin parcouru par l'œil (scanpath) lors d'une analyse approfondie de l'environnement. Dans une deuxième phase (quand on regarde la scène une deuxième fois), l'œil se contente de regarder si toutes les primitives qu'il avait observées lors de la première phase sont bien présentes.

On peut en conclure que :

- l'œil doit explorer toute la scène afin d'appréhender celle-ci dans son ensemble ;
- certains aspects de l'information requièrent une attention focalisée. Selon cette procédure attentionnelle, plusieurs éléments nécessiteront d'être traités séquentiellement ;
- il serait bon qu'une machine soit à même de réaliser un apprentissage des chemins de parcours idéaux conformément au modèle de l'œil humain.

Cette stratégie de perception de l'espace mise en œuvre par le cerveau semble très adaptée à notre problème. En effet on y trouve une alternance de phases d'exploration, de focalisation et reconstruction précise d'un objet.

2.1.2 Gestion événementielle des mouvements

L'approche la plus commune dans le processus d'exploration en vision est de déduire par interprétation des données acquises par le capteur un mouvement d'exploration plutôt que de disposer *a priori* d'un modèle permettant d'obtenir celui-ci de manière explicite.

Ce processus peut se résumer en une boucle sur l'interprétation des données dont on dispose, la prise en compte de certaines informations, et sur l'asservissement du capteur en fonction de ces informations.

[1] Petite région située au centre de la rétine. C'est dans la fovéa que la densité de photorécepteurs est la plus grande

Se basant sur les travaux de Norton et Stark [29] et reprenant la notion de "scanpath", Didday et Arbib [6] décrivent comment produire un mouvement simulant celui de l'œil en se passant de la notion d'anneau de caractéristiques (séquence ordonnée de caractéristiques perçues à chaque fixation et connue *a priori* décrite par Norton et Stark). Le but de la perception n'est alors pas de construire une représentation interne de la scène observée mais de fournir une aide à la décision : quelle action doit-on exécuter (notion de perception orientée action).

Didday et Arbib montrent comment la stratégie d'exploration peut se déduire à partir d'informations extraites des images fournies par le capteur. Ils distinguent deux zones dans l'image : la fovea ou zone de focalisation et la périphérie (celle-ci est utilisée pour maintenir une cohérence spatiale). L'utilisation de trois types de signaux (primitives de bas niveau, incohérence entre caractéristiques attendues et caractéristiques observées, signaux de confirmation d'hypothèses) permet de trouver par le calcul la localisation du point où l'attention doit se porter. Il faut noter qu'un certain poids est donné aux points situés à proximité du point de fixation courant.

Une seconde approche est l'utilisation de méthodes Bayesiennes dans le but de définir le comportement des opérateurs de vision de bas niveaux et de créer des plans d'actions. C'est l'approche utilisée par Garvey [22].

En considérant les points d'intérêt de l'image, il est possible de définir à partir d'une matrice de Markov un chemin de parcours passant par ces points [23] : la matrice décrit les probabilités de passage d'un point de fixation à un autre. On peut remarquer que ces points sont contenus dans les zones sur lesquelles il faut se focaliser. On dispose ainsi d'un chemin de parcours similaire à celui de l'œil humain. Le chemin obtenu peut être modélisé par une chaîne, chaque élément de la chaîne représentant un point d'intérêt (donc les différents points de fixation possibles). Il est possible de comparer ces chaînes, ou même de définir une distance entre deux chaînes ; on peut ainsi dire si un chemin de parcours A est plus proche du chemin B que du chemin C. Cette distance entre deux chaînes peut dépendre d'un certain nombre de paramètres en particulier de la localisation des points de focalisation dans l'image. On peut utiliser ces distances pour la reconnaissance d'objets. Il est aussi possible en utilisant cette méthode de réaliser une classification des différents points de fixation. Cette méthode fournit une séquence de focalisation proche de celle de l'œil humain, elle peut permettre de diriger les mouvements de la caméra de manière à se focaliser successivement sur chaque primitive de l'image. La limitation de cette approche est que l'on doit disposer d'une matrice de Markov adaptée à chaque scène (la taille de la matrice dépend en effet directement du nombre de points d'intérêt dans l'image et les coefficients de la matrice dépendent de la stratégie que l'on souhaite appliquer).

En fait, on peut dire que le chemin d'exploration ainsi défini dans les méthodes précédentes est le résultat d'un processus de calcul basé essentiellement sur le traitement d'image, qu'il ne peut être décrit, analysé, ou exploité pour d'autres interprétations et surtout, qu'il ne peut être disponible *a priori*. Un chemin émergent, calculé étape par étape, peut être vu comme un mélange d'asservissement visuel et de plus ou moins de "réflexion".

2.1.3 Détermination explicite des mouvements

Le but est ici de définir une stratégie de perception de l'environnement basée sur un modèle connu *a priori* et non pas en se rapportant uniquement aux informations contenues dans les images. Le système dispose ainsi d'un savoir faire qu'il exploite pour définir le chemin à parcourir.

Cette approche du problème de la perception d'une scène est très largement inspirée des travaux de Norton et Stark [29] et en particulier de la notion d'anneau de caractéristiques "feature ring". Ce processus d'exploration implique la mise en correspondance de l'anneau dont on dispose en mémoire avec l'anneau qui est construit et réactualisé à chaque mouvement de la caméra grâce aux informations de la scène.

Dans leur article, Brown et Rimey [16] proposent une approche originale du problème de la modélisation du mouvement de l'œil basée sur l'utilisation d'un modèle augmenté des chaînes de Markov cachées (AHMM).

Un modèle de Markov caché (HMM) est une généralisation d'un automate d'état fini probabiliste, et de ce fait, permet la représentation de séquences d'action à effectuer. Le système définit, dans un premier temps, les paramètres de la chaîne de Markov par apprentissage. Par la suite, il est capable de fournir au robot portant la caméra une séquence de déplacements à effectuer. On se rend compte qu'une telle approche ne peut donner de bons résultats dans la recherche d'un chemin d'exploration car le contenu de la scène n'intervient à aucun moment dans le choix du mouvement à effectuer.

Plutôt que de suivre uniquement la séquence de mouvements proposée par HMM le modèle augmenté (AHMM) présenté dans l'article possède en plus la capacité de prendre en compte des informations provenant des images fournies par la caméra et donc de s'asservir sur celles-ci.

Brown et Rimey proposent plusieurs AHMMs permettant de résoudre des problèmes différents : une chaîne modélisant des mouvements incrémentaux (les informations fournies par la AHMM sont alors $v_i \in [0, 7]$ représentant les huit directions dans lesquelles les yeux peuvent se déplacer), une chaîne modélisant de larges mouvements de l'œil lui permettant de se focaliser grossièrement et rapidement sur tel ou tel objet de la scène (les informations fournies sont alors un doublet $v_i = (\theta, R)$ où θ et R quantifient la direction et la distance relativement à la position courante).

De manière plus formelle, une chaîne de Markov est un triplet $\lambda = (A, \pi, B)$, un ensemble d'états $Q = \{q_1, \ldots, q_n\}$ et un ensemble d'actions à effectuer $V = \{v_1, \ldots, v_m\}$. La probabilité de transition à la date t de l'état q_i à l'état q_j est donnée par $A = \{a_{ij}\}$ où $a_{ij} = P(q_j \text{ à } t+1 | q_i \text{ à } t)$. Les états initiaux sont données par $\pi = \{\pi_i\}$ où $\pi_i = P(q_i \text{ à } t = 1)$. Si la HMM est dans l'état q_j, elle exécute l'action v_k en accord avec les probabilités $B = \{b_j(k)\}$ où $b_j(k) = P(v_k \text{ à } t | q_j \text{ à } t)$. On remarque que la structure du graphe des états est fondamentale à la robustesse et aux performances du système, et que sa création demande une expérimentation.

La chaîne de Markov augmentée presentée ici peut être décrite par les paramètres $\mu = (\lambda, \alpha)$ où λ représente les paramètres de la chaîne de Markov régulière et où $\alpha \in [0, 1]$ est une valeur permettant de réguler l'influence de l'asservissement de la caméra par les données. Soit $O = O_1, \ldots, O_T$ la séquence d'actions générée par la HMM, et $S = S_1, \ldots, S_T$ la séquence générée par le procesus d'asservissement. La séquence générée par la AHMM est donc $M = M_1, \ldots, M_T$ où M_i est donnée par l'équation suivante :

$$M_i = \begin{cases} S_i & \text{avec la probabilité } \alpha \\ O_i & \text{sinon} \end{cases}$$

Il faut noter que avec $\alpha = 0$ les informations contenues dans l'image ne sont pas prises en compte et avec $\alpha = 1$ on n'utilise pas les informations provenant de la chaîne de Markov.

Cette approche originale de la modélisation du mouvement de l'œil semble donner des résultats expérimentaux assez intéressants. Cependant certains aspects de la méthode peuvent sembler obscurs et mal définis, en particulier en ce qui concerne la phase de création du graphe de la AHMM.

D'autre part l'utilisation, pour l'exploration d'une scène, d'une AHMM dont les paramètres A, π et B ont été obtenus lors de l'étude d'une autre scène ne donnera vraisemblablement pas de bons résultats du fait d'une localisation différente des points d'intérêt qui caractérisent les deux environnements. Ce modèle apparaît donc assez peu adapté au cas de l'exploration d'une scène par un robot totalement autonome, même si le fait d'utiliser d'autres informations que celles contenues dans l'image soit une idée intéressante qui peut sans doute être réutilisée.

Conclusion

On peut utiliser les méthodes de recherche de chemins émergents ou explicites pour définir dans quel ordre les primitives présentes dans une image doivent être reconstruites. Cependant, ces approches, si elles peuvent donner des stratégies concernant l'ordre de reconstruction d'un certain nombre de primitives de la scène (celles qui sont vues à un instant donné), ne donnent pas une stratégie globale d'exploration.

Une deuxième étape du processus de perception globale de la scène consiste à permettre la reconstruction des primitives qui la composent, c'est à dire à acquérir des informations précises sur la géométrie de la scène.

2.2 Reconstruction 3-D

Dans cette partie, nous introduisons les différentes méthodes existantes en matière de reconstruction 3D, tout en essayant de relever les avantages ainsi que les limitations de chacune d'elles. Nous insisterons tout particulièrement sur les méthodes basées sur l'utilisation d'une caméra en mouvement sur laquelle s'est porté notre choix.

Les données tridimensionnelles d'une scène sont acquises en pratique soit directement, avec des capteurs mesurant la profondeur, soit indirectement, en exploitant par exemple les informations 2D issues d'une ou plusieurs caméras. Il existe diverses méthodes en vision par ordinateur permettant de remonter à une représentation 3D de l'environnement perçu : les approches multi-images font appel à au moins deux images pour déterminer la structure 3D d'une scène. On distingue deux catégories d'approches qui diffèrent de par la structure même du système de vision utilisé : la reconstruction 3D par stéréovision qui utilise un système de deux, voire de trois caméras et la reconstruction 3D à partir du mouvement qui utilise une seule caméra mobile. Cette approche est aussi communément appelée la vision dynamique. Dans certains cas, la caméra est en plus commandable [9], [40], [32]. On parle alors plus spécifiquement de vision active.

2.2.1 Vision Stéréoscopique

La vision stéréoscopique permet de reconstruire la structure 3D d'une scène en l'observant depuis plusieurs points de vue différents avec un système de plusieurs caméras (en général deux ou trois). Cette technique a suscité depuis de nombreuses années un grand nombre de travaux de recherche, [27], [41], [38], [42], [30]. Pour une bonne revue des méthodes existantes le lecteur pourra se referer à l'ouvrage de Ayache [7]. La stéréovision présente trois principales caractéristiques : la notion d'appariement de primitives, la contrainte épipolaire et la triangulation.

La stéréovision semble être très bien adaptée à la perception globale de l'environnement. Cependant, elle n'a plus les mêmes performances lorsqu'il s'agit d'analyser une zone bien précise de l'environnement en vue d'une reconnaissance ou d'une manipulation en robotique. Cela est dû essentiellement à la structure rigide du système de stéréovision.

Par contre, l'utilisation d'une caméra mobile est beaucoup plus souple d'emploi et ne présente aucune limitation dans le mouvement. Par ailleurs, le traitement n'étant effectué que sur une seule image contrairement à un système de stéréovision, on peut donc accéder à des cadences de traitement plus élevées pour une configuration matérielle donnée.

2.2.2 Reconstruction à partir du mouvement

L'objectif de cette classe de techniques est de réaliser la reconstruction d'une scène à partir d'un certain nombre d'images acquises par une caméra embarquée sur un robot mobile parcourant un environnement statique ou éventuellement contenant des objets en mouvement.

On peut classer les approches traitant de ce problème en deux catégories principales : les techniques continues et les techniques discrètes.

Les techniques continues qui utilisent le champ dense 2D des vitesses apparentes (optic flow) [1], [28], [36], [33], [2] qui est en général bruité et partiellement incorrect notamment près des contours d'occlusions. Ces approches sont en général très sensibles au bruit à cause du faible déplacement qu'elles imposent entre deux images successives.

Les techniques discrètes qui utilisent le déplacement dans les images de primitives particulières [3],[19], [13], [34], [20], [39], [37]. Ces techniques reposent sur trois étapes fondamentales: extraction dans la séquence d'images d'un ensemble de primitives 2D suffisamment pertinentes, appariemment des primitives sélectionnées d'image en image, reconstruction par triangulation. On retrouve ainsi le même problème que celui de la stéréovision. Il existe également des méthodes mixtes [40], [15] que nous allons à présent décrire.

2.3 Reconstruction d'un objet géométrique par vision active

La méthode que l'on se propose d'utiliser pour reconstruire les primitives 3-D est celle qui est présentée dans [14] et [15]. Cette étude est adaptée à une scène dont les objets sont statiques et où l'on dispose d'informations à priori sur la scène et sur le mouvement de la caméra.

Comme on le verra par la suite, la faible qualité des résultats obtenus en utilisant un mouvement quelconque de la caméra (vision dynamique) a rendu nécessaire l'utilisation de la vision active pour résoudre le problème.

2.3.1 Modélisation du problème de la reconstruction

On considère une caméra modélisée par son centre optique O et son plan image P_i, la distance focale de la caméra étant fixée à 1 (voir figure 1). Un point m de coordonnées $\underline{x} = (x, y, z)^T$ se projette par projection perspective en M de coordonnées $\underline{X} = (X, Y, 1)^T$ avec :

$$\underline{X} = \frac{1}{z}\,\underline{x} \tag{1}$$

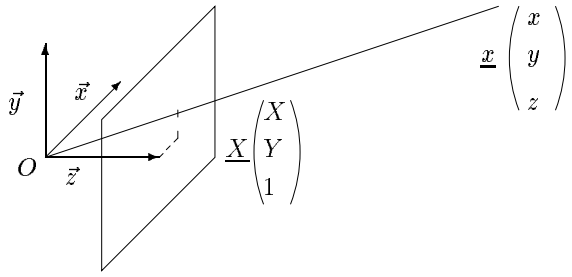


Figure 1 : Modèle de la caméra

Soit $\mathcal{P}_s$ une primitive géométrique paramétrable, elle est décrite par une équation de la forme :

$$h(\underline{x}, \underline{p}) = 0, \forall \underline{x} \in \mathcal{P}_s \tag{2}$$

où h définit la nature de la primitive et $\underline{p}$ sa configuration. Le but de la reconstruction est donc d'estimer la valeur des paramètres $\underline{p}$ afin de pouvoir reconstruire et localiser la primitive h. Il faut noter que l'on suppose connu *a priori* le type de primitive à reconstruire (droite, cylindre, sphère, etc...)

A partir de cette équation paramétrique et en utilisant le modèle de projection perspective, on peut définir les deux fonctions suivantes :

$$g(\underline{X}, \underline{P}) = 0, \forall \underline{X} \in \mathcal{P}_i \tag{3}$$

et

$$\frac{1}{z} = \mu(\underline{X}, \underline{p}_0) \tag{4}$$

La fonction $g(\underline{X}, \underline{P})$ représente la projection $\mathcal{P}_i$ dans l'image de $\mathcal{P}_s$. La fonction $\mu(\underline{X}, \underline{p}_0)$ représente la surface des limbes (voir figure 2) et exprime la relation entre les points de g et les points 3-D correspondants (les paramètres $\underline{p}_0$ sont fonctions des paramètres $\underline{p}$).

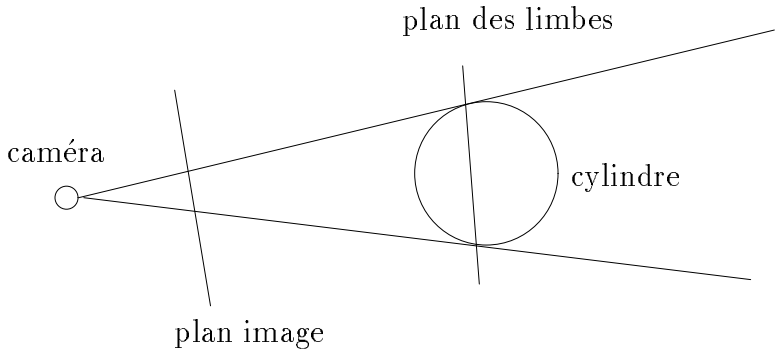


Figure 2 : Vue de dessus d'un cylindre

Soit $T = (V(O), \Omega)^T$ le torseur cinématique de la caméra où $V(O) = (V_x, V_y, V_z)^T$ représente la vitesse de translation de la caméra et $\Omega = (\Omega_x, \Omega_y, \Omega_z)^T$ sa vitesse de rotation.

La variation de $\underline{P}$ qui permet de relier le mouvement apparent de la primitive dans l'image au mouvement de la caméra s'exprime par :

$$\dot{\underline{P}} = L_{\underline{P}}^{T}(\underline{P}, \underline{p}_0)\, T \tag{5}$$

où $L_{\underline{P}}^{T}$ représente le torseur d'interaction qui caractérise les interactions entre le capteur et la primitive considérée.
Le théorème des fonctions implicites permet d'exprimer $\underline{p}_0$ en fonction des autres variables de la fonction (5) :

$$\underline{p}_0 = \underline{p}_0(T, \underline{P}, \dot{\underline{P}}) \tag{6}$$

Connaissant la structure du torseur d'interaction $L_{\underline{P}}^{T}$, la vitessse T de la caméra, les paramètres 2D $\underline{P}$ associés à $\mathcal{P}_i$ et leur variation $\dot{\underline{P}}$ dans la séquence d'images, on peut déterminer $\underline{p}_0$ qui caractérise la surface des limbes μ. Grâce aux contraintes géometriques sur le type de primitive à reconstruire (droite, cylindre,...) caractérisées par l'équation $h(\underline{x}, \underline{p}) = 0$, la connaissance de $g(\underline{X}, \underline{P})$ dans l'image permet alors de remonter, par intersection du cône de centre O et de génératrice g avec la surface des limbes aux paramètres $\underline{p}$ qui caractérisent la primitive h étudiée.

2.3.2 Reconstruction par vision active

Les résultats présentés dans [14] portant sur la reconstruction par vision dynamique de primitives de type point, droite, sphère et cylindre sont assez médiocres. En effet des erreurs de reconstruction assez importantes sont observées. Celles-ci sont dues à des erreurs de discrétisation et à une faible robustesse aux bruits. Sans stratégie relative aux mouvements de la caméra, c'est à dire avec un mouvement quelconque, cette approche est relativement inefficace. Dans le but d'améliorer ces résultats, il s'est avéré nécessaire d'aborder le problème sous un autre aspect : la vision active.

Contrairement à l'approche vision dynamique où l'on se contente d'observer le mouvement de la caméra, dans le cas de la vision active, et par analogie avec la vision humaine, le mouvement est commandé. Ce sont les informations fournies par le capteur à l'instant t qui décideront de l'emplacement de la caméra à l'instant $t+1$: le capteur est asservi.

Une analyse théorique de la vision active a été proposée dans [5],[4] et [8]. Il y est entre autre prouvé qu'un observateur actif peut résoudre des problèmes de bas niveau de manière plus efficace qu'un observateur passif. De plus, des problèmes qui sont mal posés, non linéaires et instables pour un observateur passif deviennent, pour un observateur actif, bien posés, linéaires et stables. On constate en outre que l'approche vision active est beaucoup plus robuste au bruit qu'une approche passive comme la vision dynamique. Son utilisation a donc comme but d'optimiser, en temps comme en qualité, la phase de reconstruction de la scène.

L'asservissement du capteur avec les résultats tirés des informations présentes dans les images fait de la vision active un processus de perception intelligente des données. Elle consiste en réalité à élaborer des stratégies de perception afin d'améliorer la connaissance de l'environnement.

La principale supposition de la vision active est l'inclusion de l'asservissement dans le système ainsi que la collecte des données nécessaires à cet asservissement. On verra que le modèle de l'asservissement visuel [18], [21] répond parfaitement à cette supposition.

La qualité de l'estimation des paramètres d'une primitive à partir d'une séquence d'images étant très sensible aux mouvements de la caméra, l'objectif présenté dans [14] et que nous rappelons ici est de déterminer les mouvements adéquats de celle-ci afin d'obtenir une estimation optimale des caractéristiques de la primitive.

Suppression des erreurs de discrétisation

La méthode de reconstruction dynamique décrite précédemment est basée en grande partie sur la mesure de $\underline{\dot{P}}$ (vitesse des paramètres représentant la primitive étudiée). Cette mesure de $\underline{\dot{P}}$ est impossible à effectuer exactement (car les images sont acquises à des instants discrets). On mesure seulement $\underline{\Delta P}$ représentant la variation des valeurs de $\underline{P}$ entre deux acquisitions d'images. Or l'utilisation de $\frac{\underline{\Delta P}}{\Delta t}$, dans l'équation (6) à la place de $\underline{\dot{P}}$ entraine des erreurs dans l'estimation des paramètres $\underline{p}_0$ qui se répercutent dans l'estimation des paramètres $\underline{p}$.

Il faut donc contraindre les mouvements de la caméra de manière à assurer que :

$$\underline{\dot{P}} = \frac{\underline{\Delta P}}{\Delta t}, \forall t \tag{7}$$

Une condition générale et suffisante pour respecter la condition (7) est de contraindre les mouvements de la caméra de telle sorte que :

$$\underline{\dot{P}} = \underline{\dot{p}}_0 = 0, \forall t \tag{8}$$

Cette équation signifie que l'image et l'équation de la surface des limbes de la primitive doivent rester constantes le long de la trajectoire de la caméra.

Minimisation des effets des erreurs de mesure

La configuration d'une primitive dans l'image influe sur la qualité de son estimation. Il est ainsi montré dans [14] que certaines contraintes sur la position de la primitive dans l'image ainsi que des contraintes sur le mouvement de la caméra sont nécessaires à une bonne estimation des paramètres de h. Par exemple dans le cas d'un cylindre, celui-ci doit apparaitre centré et vertical ou horizontal.

Pour respecter les contraintes nécessaires à la suppression des erreurs de discrétisation et celles nécessaires à la minimisation des effets des erreurs de mesure, il faut définir un modèle permettant de générer les mouvements de la caméra à partir des informations présentes dans les images tout en tenant compte des contraintes presentées ci-dessus : la méthode d'asservissement visuel décrite à présent permet d'effectuer automatiquement ces différents mouvements.

2.3.3 Asservissement des mouvements de la caméra

On souhaite donc pouvoir générer automatiquement les mouvements de la caméra en fonction de certaines contraintes. Dans cette optique deux approches utilisant les informations visuelles sont à distinguer :

- l'asservissement en situation de la caméra par rapport à son environnement ;
- l'asservissement visuel basé sur une régulation dans l'image.

Dans la première approche, l'asservissement en situation, les tâches à réaliser (positionnement par rapport à un objet) sont exprimées en fonction d'une situation $\bar{r}^*$ (position et orientation) à atteindre entre la caméra et cet objet. Pour atteindre la situation voulue, une estimation $\hat{\bar{r}}$ de la situation

courante est alors nécessaire. Cette approche nécessite une phase d'interprétation des informations visuelles de manière à estimer cette situation courante. Cette phase d'interprétation pose deux problèmes :

- les algorithmes mis en œuvre peuvent être très lourds, rendant difficile une interprétation en temps réel.
- les erreurs de précision, inhérentes à cette interprétation, entrainent des erreurs dans l'estimation de la situation courante, ce qui peut entrainer une convergence de l'algorithme vers une situation qui n'est pas celle souhaitée.

L'objectif de la deuxième approche, l'asservissement visuel, est d'introduire directement les informations extraites de l'image dans la boucle de commande, évitant ainsi la phase d'interprétation de ces informations ; cette approche semble donc optimale au niveau du temps de calcul.

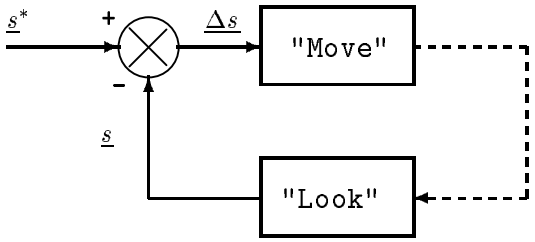


Figure 3 : Asservissement visuel

Dans cette approche, les consignes sont exprimées sous la forme d'un motif à atteindre dans l'image. Si $\underline{s}^*$ est le motif final désiré dans l'image, les informations visuelles $\underline{s}$ choisies pour constituer ce motif doivent converger vers $\underline{s}^*$ (voir figure 3).

Par exemple, si dans une image, on dispose d'un segment de droite caractérisé par ses paramètres ρ et θ ($x\cos\theta + y\sin\theta - \rho = 0$), on peut vouloir que ce segment apparaisse vertical centré dans l'image. Le motif final désiré dans l'image est donc caractérisé par $\underline{s}^* = (\underline{\rho}^*, \underline{\theta}^*) = (0, \frac{\pi}{2})$ et les informations visuelles sont dans ce cas les ρ et θ mesurés. L'asservissement visuel doit permettre le calcul automatique des mouvements de la caméra par une loi de commande en boucle fermée de manière à minimiser l'erreur $(\underline{s} - \underline{s}^*)$ dans l'image.

Une modélisation du problème est donnée dans [18]. La caméra est embarquée sur l'effecteur d'un robot. Les déplacements de la caméra sont donc réalisés à l'aide des différents axes constituant le robot et la situation de la caméra ne dépend donc que de la valeur des coordonnées articulaires $\underline{q}$. De manière générale, si les objets observés sont eux-mêmes mobiles, les informations visuelles choisies pour représenter les tâches $\underline{s}$ peuvent s'écrire :

$$\underline{s} = \underline{s}(\underline{q}, t) \tag{9}$$

où la variable temporelle t permet la prise en compte du mouvement des objets.

On définit alors une fonction de tâche $\underline{e}(\underline{q}, t)$ qui exprime une erreur entre une configuration particulière désirée de primitives dans l'image et la configuration courante observée. On a :

$$\underline{e}(\underline{q}, t) = C(\underline{s}(\underline{q}, t) - \underline{s}^*) \tag{10}$$

où C est une matrice dite de combinaison. C peut être considérée comme le jacobien inverse de la tâche et est obtenue en choisissant $C = L_{\underline{s}}^{T^+}$ où $L_{\underline{s}}^{T}$, le torseur d'interaction associé à $\underline{s}$, est définie par l'équation (5).

On montre qu'un certain nombre d'informations visuelles peuvent être utilisées dans cette approche. Deux conditions sont cependant nécessaires pour permettre cette utilisation :

- l'existence d'algorithmes de traitement d'images capables de mesurer cette information à cadence rapide ;
- la possibilité de calculer de manière explicite le torseur d'interaction de l'information (c'est-à-dire le lien entre les variations dans l'image des informations visuelles et les mouvements de la caméra).

Dans le cas qui nous intéresse, la reconstruction 3-D de primitives, une commande en boucle fermée sur des informations visuelles (telle que l'asservissement visuel) paraît parfaitement adaptée pour générer automatiquement les mouvements nécessaires à la reconstruction.

Afin d'obtenir une meilleure reconstruction des primitives géométriques, des contraintes sur ces mouvements sont introduites. En effet, pour supprimer les erreurs de discrétisation, ceux-ci doivent être tels que $\underline{\dot{P}} = 0, \forall t$, ce qui peut s'exprimer par $\underline{P}(t) - \underline{P}_d = 0, \forall t$ où $\underline{P}(t)$ est la valeur mesurée à chaque itération des paramètres de la primitive et où $\underline{P}_d$ est la valeur de $\underline{P}$ mesurée à la première itération ou à l'itération précédente. De même, pour minimiser les erreurs de mesure, les mouvements doivent assurer $\underline{P}(t) - \underline{P}_d = 0, \forall t$ où $\underline{P}_d$ est ici la configuration optimale choisie de la primitive dans l'image (par exemple pour un cylindre qu'il soit centré et vertical dans l'image).

Dans les deux cas, la fonction de tâche correspondante s'exprime sous la forme :

$$\underline{e} = W^+ C(\underline{P}(t) - \underline{P}_d) + (I - W^+ W)\underline{e}_2 \tag{11}$$

où

- $\underline{e}$ est la fonction que l'on souhaite réguler à zéro
- $\underline{e}_2$ est une tâche secondaire (ici un suivi de trajectoire) construite afin de permettre des mouvements de la caméra tout en assurant la contrainte $\underline{P}(t) - \underline{P}_d = 0, \forall t$;
- I représente l'identité ;
- W^+ et $I - W^+W$ sont des opérateurs de projection qui assurent la compatibilité de la tâche $\underline{e}_2$ avec la contrainte $\underline{P}(t) = \underline{P}_d$.

Il est en effet indispensable d'avoir des mouvements de la caméra pour estimer la structure des primitives (dans le cas d'un cylindre, ce mouvement consistera à tourner autour du cylindre tout en l'observant centré et vertical dans l'image). C'est cette tâche secondaire qui permet d'effectuer ces mouvements et d'assurer les contraintes supplémentaires, à savoir $\underline{\dot{p}_0} = 0$ pour supprimer les erreurs de discrétisation et celles qui fournissent une estimation plus robuste.

Les résultats de reconstruction 3-D obtenus dans [14] en utilisant l'asservissement visuel sont de bien meilleure qualité que ceux obtenus par l'approche vision dynamique. Un problème se pose cependant, cette méthode de reconstruction ne permet d'estimer les paramètres que d'une primitive à la fois.

3 Reconstruction d'une scène tridimensionnelle

La méthode de reconstruction choisie ne permet pas de reconstruire plusieurs primitives simultanément, il faut donc enchaîner des phases de focalisation et de reconstruction d'une primitive. Si l'on suppose que l'on sait reconstruire chaque primitive de la scène indépendamment l'une de l'autre, les problèmes posés sont donc les suivants : comment repérer les objets dans la scène ? Sur quel objet doit-on se focaliser et comment passer au suivant ?

La stratégie d'exploration globale de la scène sera donc constituée :

- de phases de sélection des primitives à reconstruire. Il s'agit ici de décider à partir des différentes informations contenues dans l'image (présence d'une ou de plusieurs primitives) laquelle va être reconstruite (sur quelle primitive doit-on se focaliser?) En cas de conflit entre plusieurs primitives, un poids plus important peut par exemple être donné aux primitives situées à proximité du point de focalisation courant ;

- de phases d'asservissement visuel durant lesquelles la primitive 3-D sur laquelle la caméra est focalisé sera reconstruite de manière optimale. Parallèlement, on reconstruira de manière sommaire, par vision dynamique, les autres primitives de la scène ;

- de phases de recherche dans l'environnement d'informations permettant de sélectionner les futures primitives à reconstruire (exploration). Dans le cas où l'on ne détecterait dans l'image aucune primitive 2D se rapportant à un objet 3-D non reconstruit, il apparaît nécessaire de faire effectuer à la caméra une tâche d'exploration ;

- de phases de mise à jour du modèle représentant la scène 3-D. La reconstruction de la scène se fait de manière incrémentale. Au début de la phase de reconstruction les connaissances dont on dispose sur la scène sont presques nulles (on sait simplement que les primitives recherchées sont des cylindres et des droites), mais la reconstruction de chaque primitive 3-D entraîne une augmentation des connaissances sur la structure de la scène. Ces connaissances peuvent être utilisées soit lors de la phase d'exploration, soit lors du choix de la prochaîne primitive à reconstruire.

On propose donc dans ce paragraphe une méthode pour enchaîner les reconstructions basées sur les informations présentes dans l'image ainsi que sur les données déjà acquises sur l'organisation spatiale de la scène. On présente aussi un cadre général au problème de l'exploration en tenant compte de l'objectif de complétude qui doit être atteint pour la reconstruction de la scène. On donne enfin un algorithme permettant d'assurer l'enchaînement de ces différentes tâches.

3.1 Une reconstruction incrémentale de la scène

Avant de proposer un cadre général à la reconstruction 3-D d'une scène, il apparaît nécessaire de préciser quelques notations et de définir ce qu'est une reconstruction et pourquoi il apparaît necéssaire de reconstruire la scène de manière incrémentale.

3.1.1 Notations

Position de la caméra Soit $\mathcal{C}$ l'ensemble des positions accessibles par la caméra. On notera $\phi_t \in \mathcal{C}$ la position de la caméra à un instant t donné.
De même, on notera Φ_I l'ensemble des positions occupées par la caméra pendant un intervalle de temps I :

$$\Phi_I = \{\phi_t \ , \ t \in I\}$$

On notera Ω la scène 3-D étudiée. On considérera, sans perte de généralité dans le cas qui nous intéresse, que la scène à étudier est comprise dans un sous espace de $\mathbb{R}^3$ borné.

Structure de la scène étudiée (voir figure 4)

On peut décrire une scène comme un ensemble d'objets. Dans le cas qui nous intéresse, on considérera que les éléments de base de la scène sont des primitives géométriques paramétrables (droites, cylindres, sphères,...). Soit $\mathcal{P}_S$ une de ces primitives. Rappelons qu'elle peut être décrite par une équation de la forme :

$$h(\underline{x}, \underline{p}) = 0, \forall \underline{x} \in \mathcal{P}_s$$

où h représente la nature de la primitive et $\underline{p}$ sa configuration.

On notera ω_{ϕ_t} ou ω_ϕ le sous-ensemble des objets de la scène Ω observés par la caméra depuis la position ϕ_t. De même on notera ω_{Φ_I} le sous ensemble des objets de la scène Ω observés depuis les positions Φ_I occupées par la caméra pendant l'intervalle de temps I.

Pour une position ϕ_t de la caméra, la partie ω_{ϕ_t} de la scène observée peut être définie comme la réunion de toutes les primitives observables depuis cette position :

$$\omega_{\phi_t} = \{\mathcal{P}_{S_i} / \mathcal{P}_{S_i} \text{ est visible depuis } \phi_t\}$$

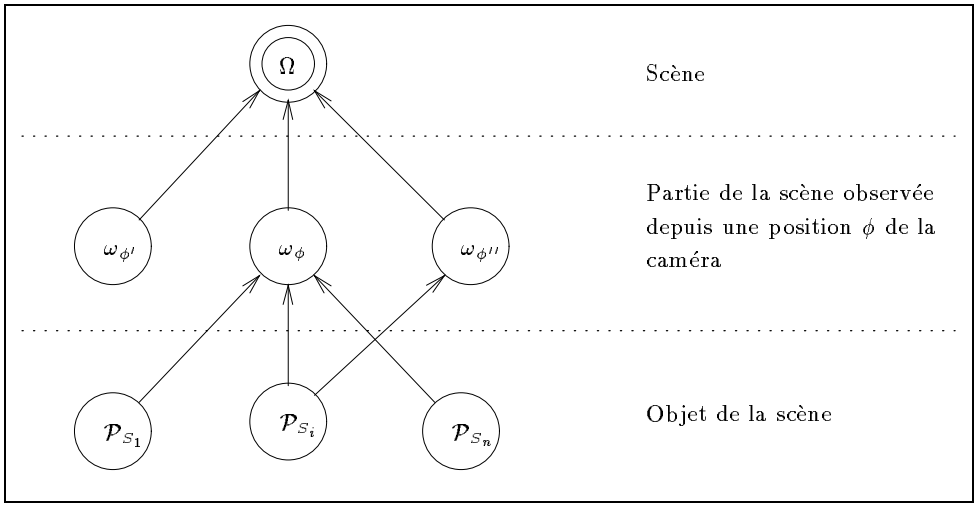


Figure 4 : Structure d'une scène

Une même primitive géométrique $\mathcal{P}_S$ peut appartenir à deux sous ensembles ω_ϕ différents.

Exemple Soit Ω la scène répresentée par la figure 5 (on considère ici une scène Ω composée uniquement de cylindres verticaux) Le cône de sommet C représente la partie de la scène observée par la

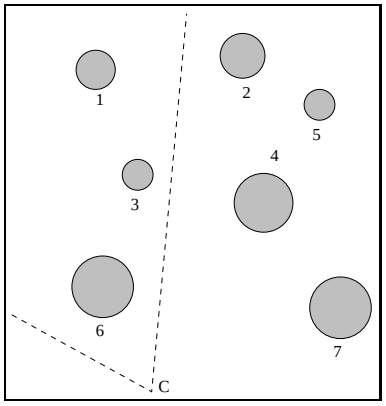


Figure 5 : exemple d'une scène vue de dessus

caméra à la position ϕ_C. L'ensemble ω_{ϕ_C} des primitives observables depuis cette position ϕ_C est donc donné par :

$$\omega_{\phi_C} = \{\mathcal{P}_{S_3}, \mathcal{P}_{S_6}\}$$

Dans ce cas, on peut remarquer que $\mathcal{P}_{S_1}$ est dans le cône de vision de la caméra mais est occultée par $\mathcal{P}_{S_6}$ et $\mathcal{P}_{S_3}$.

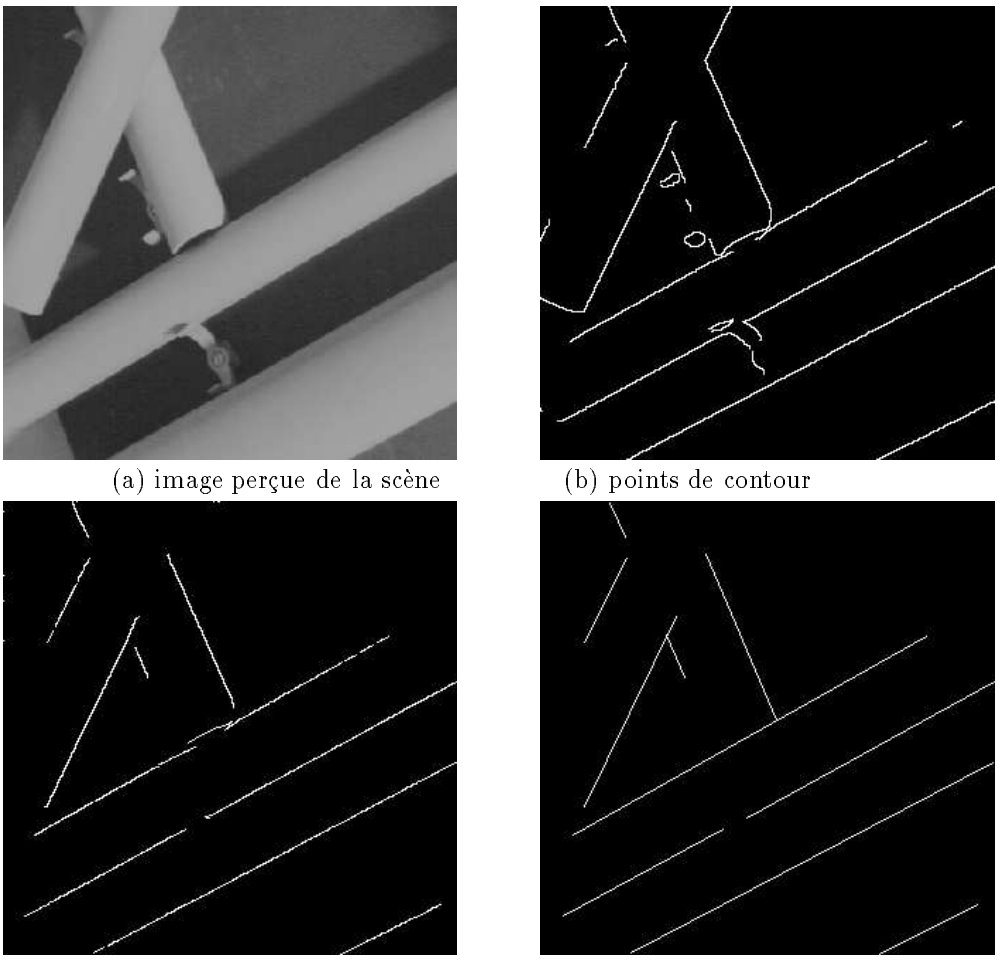
3.1.2 Reconstruction incrémentale

3.1.2.1 Information perçue Le but de la reconstruction est d'obtenir une information aussi précise et aussi complète que possible du monde Ω. Les seules informations disponibles sur Ω sont extraites des images acquises par la caméra.

A partir d'une image en niveau de gris (voir figure 6a) acquise par la caméra, on effectue une série de prétraitements (en l'occurence, extraction des points de contraste et transformée de Hough) de manière à disposer d'une information utilisable. Dans le cas qui nous intéresse cette information sera constituée d'une liste de segments repérés par leur position dans l'image (voir figure 6d) que l'on notera $p(\omega_\phi)$. $p(\omega_\phi)$ représente l'information extraite de la projection de chaque primitive de l'ensemble ω_ϕ dans le plan image.

L'ensemble des informations captées par la caméra pendant un intervalle de temps I est défini par :

$$p(\omega_{\Phi_I}) = \{p(\omega_{\phi_t}) \ , \ \phi_t \in \Phi_I\}$$



(a) image perçue de la scène (b) points de contour

(c) points retenus (d) segments retenus

Figure 6 : Création d'une base de données

3.1.2.2 Création d'une carte Etant donné un ensemble d'informations sur un monde Ω, on définira une carte comme étant la représentation la plus complète possible de ce monde à la vue des seules informations disponibles. On notera cette carte $S(\Omega)$.

Soit ω_ϕ la partie du monde Ω observée depuis une position ϕ de la caméra. L'objectif d'une reconstruction est d'obtenir une carte $S(\omega_\phi)$ de cette partie du monde. Soit $I_\mathcal{R}$ l'intervalle de temps pendant lequel a lieu cette reconstruction. On définit alors $\Phi_\mathcal{R}$ comme étant l'ensemble des positions occupées par la caméra pendant cet intervalle.

Soit $p(\omega_\phi)$ les informations acquises sur une partie ω du monde Ω à partir d'une position ϕ initiale de la caméra. On notera $\mathcal{R}_S$ la fonction reconstruction qui, à partir de cet ensemble d'informations $p(\omega_\phi)$ et des informations complémentaires acquises pendant le processus de reconstruction $p(\omega_{\Phi_\mathcal{R}})$, permet de créer une carte partielle de l'environnement.

Dans le cas où ω_ϕ représente un ensemble de n primitives indépendantes : $\omega = \{\mathcal{P}_{S_1}, \cdots, \mathcal{P}_{S_n}\}$ où chaque $\mathcal{P}_{S_i}$ représente une primitive, alors la reconstruction de ω est une succession de plusieurs reconstructions indépendantes. La méthode de reconstruction par vision active choisie ne permet pas de reconstruire plusieurs primitives géométriques en même temps, il est donc nécessaire de les reconstruire les unes après les autres. La carte partielle de la zone observée depuis la position ω_ϕ sera donc créée de manière incrémentale. Ainsi, on peut interpréter cette carte comme étant la succession des reconstructions de chacune des primitives de la scène observée :

$$\mathcal{R}_S(p(\omega_\phi)) = \sum_{i=1}^{n} \mathcal{R}_S(p(\mathcal{P}_{S_i})) \quad \text{où} \quad \omega_\phi = \{\mathcal{P}_{S_i} / i = 1 \ldots n\}$$

De même, la carte de la scène $S(\Omega)$ sera elle-aussi construite de manière incrémentale, c'est à dire que l'on reconstruira successivement des ensembles ω_ϕ de primitives. Quand un ensemble de reconstructions sera terminée, $S(\Omega)$ sera incrémentée avec les paramètres des primitives reconstruites. La carte $S(\Omega)$ peut être définie par :

$$S(\Omega) = \sum_{\omega \in \Omega} \mathcal{R}_S(p(\omega_\phi))$$

Il faudra s'assurer que la reconstruction est complète (*i.e* aucune primitive n'a été oubliée). Dans ce cas où ω représente un ensemble de primitives, il faut déterminer dans quel ordre ces primitives doivent être reconstruites, ce que nous décrivons à présent.

3.2 Focalisation sur une primitive

3.2.1 Gestion de la base de données 2-D

Comme nous l'avons décrit précédement, les informations 2-D $p(\omega_\phi)$ sont constituées d'un ensemble de segments de droites. Pour chacun d'entre eux, on dispose de la valeur de ses paramètres polaires ρ et θ, de sa longueur en nombre de pixels, ainsi que de la liste des pixels qui composent ce segment.

Cette base de données va être utilisée pour déterminer l'ordre de reconstruction des primitives de ω. Elle sera mise à jour au fur et à mesure de la reconstruction et un segment sera supprimé de $p(\omega_\phi)$ lorsqu'il aura été utilisé pour la reconstruction d'une primitive.

Quelques problèmes se posent cependant concernant le contenu de cette base. Une même primitive $\mathcal{P}_s$ peut en effet se projeter dans l'image suivant plusieurs segments en raison de problèmes de segmentation ou d'occultation. Il faudra donc veiller, à l'aide des reconstructions sommaires décrites §3.3.4, à ne pas reconstruire plusieurs fois la même primitive.

C'est à partir de cette base de données que l'on va déterminer dans quel ordre seront reconstruits les différentes primitives (droites et cylindres) de la scène.

3.2.2 Stratégie de focalisation

On suppose que les segments présents dans la base de données sont numérotés de 1 à n. On peut utiliser, pour déterminer quel segment sera à la base de la prochaine reconstruction, une matrice de transition A où les coefficients a_{ij} représentent la probabilité de passage du segment i au segment j.

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ \vdots & a_{ij} & \vdots \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

Les propriétés de la matrice A sont les suivantes :

$$\forall i \in [1, n], \sum_{j=1}^{n} a_{ij} = 1$$

$$\forall i \in [1,n],\ a_{ii} = 0$$

Les coefficients sont déterminés en fonction de la stratégie que l'on décide de mettre en œuvre : ils dépendent donc des caractéristiques des segments présents dans la base de données, c'est à dire, par exemple, de leur taille, de leur luminosité et de leur position dans l'image. Ces caractéristiques vont permettre d'affecter à chaque segment un certain poids : on notera $P(i)$ le poids du segment i.

Deux cas sont envisageables concernant la valeur des coefficients :

- les transitions sont définies de manière purement déterministe :

$$\forall (i,j) \in [1,n] \times [1,n], \quad a_{ij} = 1 \text{ ou } a_{ij} = 0$$

- les transitions peuvent être définies de manière probabiliste : $a_{ij} < 1$, on aboutit dans ce cas à des mouvements de la caméra qui seraient plus proches de ceux de l'œil humain [23].

  On peut alors par exemple, calculer la valeur de chaque coefficient par la formule suivante (en supposant que $\sum_i P(i) = 1$) :

$$a_{ij} = \begin{cases} \frac{P(j)}{1-P(i)} & \text{si } i \neq j \\ 0 & \text{sinon} \end{cases}$$

Après chaque reconstruction d'une primitive, le robot se replacera à sa position de départ, de manière à observer la scène correspondant à la base de données en mémoire, et reconstruira la primitive suivante. Le segment que l'on vient de traiter est alors ôté de la base.

Une phase d'exploration sera déclenchée quand tous les segments de la base de données seront associés à une primitive de la scène, c'est à dire quand la base sera vide. Cette phase d'exploration est décrite au §3.4.

3.2.3 Calcul du poids de chaque segment

Soit $P(i)$ le poids de chaque segment, ce poids dépend de plusieurs paramètres :

- sa position dans l'image et décrite par les paramètres ρ et θ ;
- sa longueur l.

On peut vouloir reconstruire d'abord les longs segments plutôt que ceux qui sont plus courts, ou bien favoriser les segments verticaux ou horizontaux, ou encore prendre en premier les segments situés plus ou moins loin du point de fixation courant. Comme ce sont les poids $P(i)$ de chaque segment qui vont déterminer l'ordre de reconstruction, il faut prendre en compte dans le calcul de $P(i)$ tous ces choix de l'utilisateur.

On peut proposer une méthode pour calculer $P(i)$ qui tienne compte de la stratégie que l'on souhaite mettre en œuvre. Considérons P_i le poids accordé à la position du segment i dans l'image et L_i sa longueur. On posera le poids total $P(i)$ de i comme étant une combinaison linéaire de P_i et L_i :

$$P(i) = \frac{\alpha P_i + \beta L_i}{\alpha + \beta}$$

où α et β caractérisent l'importance relative accordée à la position et à la longueur dans le calcul du poids.

Poids accordé à la longueur Soit l_i une valeur caractérisant la longueur du segment i. On considère que $l_i \in [0,1]$ on calculant l_i de la manière suivante : $l_i = \frac{l'_i}{l_{max}}$ où l'_i est la longueur du segment i en nombre de pixels et où l_{max} est la longueur en nombre de pixels du plus grand segment de la base de données $p(\omega_\phi)$.

On pose $L_i = L_g(l_i, c)$ tel que, pour :

- $c > 0$, on favorise les grands segments ;
- $c < 0$, on favorise les petits segments ;

- $c = 0$, la longueur n'intervienne pas dans le calcul du poids du segment.

Une fonction $L_g(l_i, c)$ correspondant à cette spécification est par exemple donnée par :

$$L_g(l_i, c) = \begin{cases} |c\, l_i| & \text{si } c \geq 0 \\ |c\,(1 - l_i)| & \text{si } c < 0 \end{cases}$$

Poids accordé à la position Outre sa longueur, le segment i est caractérisé par sa position dans l'image. Rappelons que l'équation de ce segment est donnée par : $x \cos\theta + y \sin\theta = \rho$.

Comme pour le poids accordé à la longueur, on définit le poids accordé à la position par $P_i = P(\rho, a_1, \theta, a_2)$ tel que :

- a_1 caractérise la stratégie concernant l'inclinaison θ. On a $a_1 \in [-1, 1]$ tel que si :
  - $a_1 \in [-1, 0[$, on favorise les segments horizontaux ;
  - $a_1 \in]0, 1]$, on favorise les segments verticaux ;
  - $a_1 = 0$, on ne tient pas compte de l'inclinaison ;
- a_2 caractérise la stratégie concernant la distance au point de fixation courant ρ. On a $a_2 \in [-1, 1]$ tel que si :
  - $a_2 \in [-1, 0[$ on favorise les segment éloignés du point de fixation courant ;
  - $a_2 \in]0, 1]$ on favorise les segment proches du point de fixation courant ;
  - $a_2 = 0$ on ne tient pas compte de la distance au point de fixation courant

Une fonction $P(\rho, a_1, \theta, a_2)$ qui correspond à cette spécification est la suivante :

$$P(\rho_i, a_1, \theta_i, a_2) = \Theta(a_1, \theta_i) + \rho(a_2, \rho_i)$$

où

$$\Theta(a_1, \theta) = \begin{cases} |a_1 \cos\theta_i| & \text{si } a_1 \geq 0 \\ |a_1 \sin\theta_i| & \text{si } a_1 < 0 \end{cases} \quad \text{et} \quad \rho(a_1, \rho) = \begin{cases} a_2(1 - \rho_i) & \text{si } a_2 \geq 0 \\ -a_2\, \rho_i & \text{si } a_2 < 0 \end{cases}$$

où ρ_i est ici une valeur normalisée de la distance à l'origine ($\rho_i \in [0, 1]$).

3.2.4 Mise à jour des poids

Cette méthode pour calculer les poids accordés à chaque segment ne peut constituer qu'une inititialisation. En effet, au fur et à mesure que la scène sera reconstruite, les informations sur le monde Ω seront de plus en plus précises et influeront directement sur l'ordre de reconstruction.

L'objectif pourrait par exemple être, pour le robot, de minimiser la longueur du chemin à parcourir lors du processus de reconstruction. Soit ω la partie du monde Ω observée, dont les segments sont répertoriés dans la base de donnée 2-D. ω contient n primitives géométriques paramétrables : $\omega = \{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_n\}$. On définit un chemin de parcours comme étant une permutation sur l'ensemble ω : $\omega_\sigma = \{\mathcal{P}_{\sigma(1)}, \ldots, \mathcal{P}_{\sigma(i)}, \ldots, \mathcal{P}_{\sigma(n)}\}$

Le but est donc de trouver la permutation σ qui minimise la distance totale parcourue par la caméra sous la contrainte de reconstruire toutes les primitives de ω :

$$\sigma = \arg\min_\sigma \sum_{i=1}^{n-1} d(\mathcal{P}_{\sigma(i)}, \mathcal{P}_{\sigma(i+1)})$$

où $d(\mathcal{P}_i, \mathcal{P}_j)$ est la distance entre deux primitives.

Signalons cependant que, si les paramètres de la primitive $\mathcal{P}_i$ sont connus avec précision (puisqu'il viennent d'être estimés), les paramètre de la primitive $\mathcal{P}_j$ ne sont eux, connus que de manière approchée (ils sont estimés grossièrement par une reconstruction sommaire (cf §3.3.4).

Le problème de la minimisation d'une telle distance correspond à un problème classique d'algorithmique des graphes :

On peut représenter l'ensemble des primitives ω par un graphe valué G tel que $G = (X, \Gamma, v)$ où

- X représente l'ensemble des nœuds du graphe et correspond à l'ensemble ω :

$$X = \omega = \{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_n\}$$

- Γ représente l'ensemble des transitions, toutes les primitives étant reliées deux à deux :

$$\forall \mathcal{P}_i, \mathcal{P}_j \in \omega, \mathcal{P}_i \neq \mathcal{P}_j, (\mathcal{P}_i, \mathcal{P}_j) \in \Gamma$$

- v est une fonction définie sur Γ à valeur dans $\mathbb{R}$

$$\forall i, \forall j, i \neq j, v(i,j) = d(\mathcal{P}_i, \mathcal{P}_j)$$

Ce problème consistant à trouver le chemin le plus court passant par tous les nœuds du graphe est équivalent à celui du "voyageur de commerce".

La mise à jour des poids $P(i)$ de chaque segment doit donc pouvoir être effectuée en fonction de la permutation ω_σ optimale au sens de la longueur du chemin ainsi défini. Nous ne décrirons pas davantage la recherche de cette permutation puisqu'elle constitue l'une des perspectives de ce travail.

3.3 Reconstruction d'une primitive : le cas du cylindre

Dans le cas du cylindre, nous diviserons la reconstruction en trois étapes :

- La méthode de reconstruction décrite dans [14] et rappelée dans le §2.3.2 permet d'obtenir, en utilisant la projection d'un seul limbe, le rayon, les paramètres de position et les composantes du vecteur directeur de l'axe du cylindre. On appelera $\mathcal{R}_p$ cette phase de la reconstruction.
- On calcule ensuite la position dans l'image du deuxième limbe afin de le supprimer de la base de données et de faire une reconstruction du cylindre basée sur les deux limbes. On appelera $\mathcal{R}_v$ cette phase de la reconstruction.
- Il faut enfin définir un dernier paramètre : la longueur du cylindre. On notera $\mathcal{R}_l$ cette phase de la reconstruction.

3.3.1 Estimation des paramètres d'un cylindre

Un cylindre peut être représenté par une équation du type :

$$h(\underline{x}, \underline{p}) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - (ax + by + cz)^2 - r^2 = 0 \tag{12}$$

où les paramètres $\underline{p}$ sont :

- r, le rayon du cylindre,
- a, b et c, les composantes de son axe d'orientation $\vec{V}$,
- $m_0 = (x_0, y_0, z_0)$, un point de l'axe du cylindre.

De manière à ce que ce paramétrage complet devienne minimal, on rajoute les deux contraintes suivantes :

$$a^2 + b^2 + c^2 = 1 \tag{13}$$
$$ax_0 + by_0 + cz_0 = 0 \tag{14}$$

La deuxième contrainte implique que le point m_0 est le point de l'axe le plus proche du centre optique (voir figure 7).

On a vu que pour reconstruire une primitive géométrique de manière optimale, deux conditions étaient nécessaires. Dans le but de supprimer les erreurs de discrétisation il faut contraindre les mouvements de la caméra de telle manière que $\dot{\underline{P}} = 0$, ce qui signifie que la projection de la primitive doit rester constante tout au long de la séquence d'images.

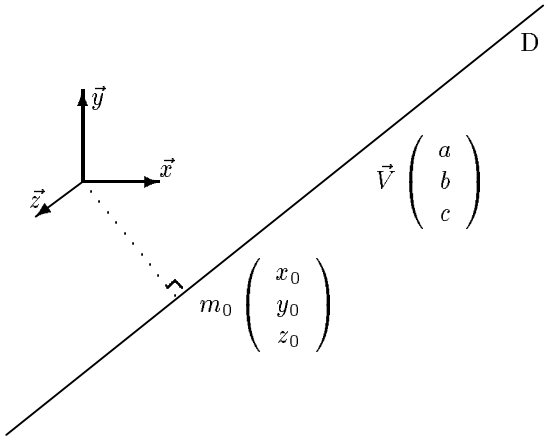


Figure 7 : paramètres du cylindre

D'autre part, afin de minimiser les effets des erreurs de mesure, il faut minimiser l'incertitude σ_p sur l'estimation des paramètres p. Ceci implique des contraintes sur le mouvement de la caméra par rapport à la primitive et sur la position de la primitive dans l'image.

Dans le cas particulier du cylindre, il est montré dans [14] que l'estimation des paramètres d'un cylindre est optimale quand celui ci se projette en deux droites symétriques verticales dans l'image tel que $\rho_1 = -\rho_2$ et $\theta_1 = \theta_2 = k\frac{\pi}{2}$ puis d'effectuer un mouvement de rotation autour de l'axe du cylindre.

Cependant, il apparait impossible de reconstruire directement un cylindre à partir de ses deux limbes. En effet, sans la connaissance *a priori* de ses paramètres, on ne peut pas extraire de la base de données 2-D les deux limbes d'un même cylindre. Il faut donc préalablement faire une estimation des paramètres du cylindre basée sur un seul limbe (il convient alors de maintenir celui-ci vertical centré dans l'image et de tourner autour). L'estimation des paramètres étant réalisée, on peut alors effectuer une reconstruction optimale du cylindre basée sur ses deux limbes et qui permet d'avoir une meilleure estimation des paramètres.

D'autre part, si le segment qui a servi de base à la reconstruction ne correspond pas à un limbe d'un cylindre mais à la projection d'une droite, l'estimation des paramètres présentera un cylindre de rayon nul. Dans ce cas particulier, l'étape suivante de reconstruction basée sur deux limbes sera remplacée par une estimation des paramètres de la droite à partir d'un module spécialement conçu à cet effet [14] et donnant des résultats plus robuste.

3.3.2 Reconstruction basée sur les deux limbes

Le but de cette partie de la reconstruction est tout d'abord de vérifier si la projection du deuxième limbe du cylindre est bien présente dans l'image. Il faut donc calculer sa position estimée à partir des paramètres du cylindre que l'on vient de déterminer. Ensuite, dans le cas où le deuxième limbe ne serait pas présent dans l'image courante, on asservit la caméra de manière à aller vérifier sa présence.

L'objectif de cette vérification est de supprimer de la base de données 2D le segment correspondant à la projection du deuxième limbe du cylindre dans le but d'éviter de reconstruire deux fois le même cylindre. De plus, pour améliorer la précision du calcul, une reconstruction basée sur la projection des deux limbes est effectuée [14].

Calcul Connaissant les paramètres $\underline{p}$ du cylindre, il est facile de retrouver les équations de chacun des limbes [18].

En appliquant les équations de projection perspective (1) sur l'équation (12), on obtient un polynôme du second degré en $1/z$:

$$A\,\frac{1}{z^2} - 2\,B\,\frac{1}{z} + C = 0 \tag{15}$$

$$\text{avec} \begin{cases} A = x_0^2 + y_0^2 + z_0^2 - r^2 \\ B = x_0 X + y_0 Y + z_0 \\ C = X^2 + Y^2 + 1 - (aX + bY + c)^2 \end{cases}$$

Ce polynôme signifie que pour un point quelconque (X, Y) de l'image, ce point appartient à la projection des limbes du cylindre si le discriminant Δ de (15) est nul. Après quelques développements, on peut factoriser Δ sous la forme :

$$\Delta = [rB/\sqrt{A} - (\alpha X + \beta Y + \gamma)][rB/\sqrt{A} + (\alpha X + \beta Y + \gamma)] = 0 \tag{16}$$

$$\text{avec} \begin{cases} \alpha = cy_0 - bz_0 \\ \beta = az_0 - cx_0 \\ \gamma = bx_0 - ay_0 \end{cases}$$

qui donne l'équation de deux droites (une seule si le rayon du cylindre est nul). En utilisant la représentation (ρ, θ) des droites, on obtient:

$$\begin{array}{rcl} D_1 & : & X\cos\theta_1 + Y\sin\theta_1 - \rho_1 = 0 \\ D_2 & : & X\cos\theta_2 + Y\sin\theta_2 - \rho_2 = 0 \end{array} \tag{17}$$

$$\text{avec} \begin{cases} \cos\theta_1 = \frac{rx_0/\sqrt{A}-\alpha}{\sqrt{(rx_0/\sqrt{A}-\alpha)^2+(ry_0/\sqrt{A}-\beta)^2}}, & \cos\theta_2 = \frac{rx_0/\sqrt{A}+\alpha}{\sqrt{(rx_0/\sqrt{A}+\alpha)^2+(ry_0/\sqrt{A}+\beta)^2}} \\ \sin\theta_1 = \frac{ry_0/\sqrt{A}-\beta}{\sqrt{(rx_0/\sqrt{A}-\alpha)^2+(ry_0/\sqrt{A}-\beta)^2}}, & \sin\theta_2 = \frac{ry_0/\sqrt{A}+\beta}{\sqrt{(rx_0/\sqrt{A}+\alpha)^2+(ry_0/\sqrt{A}+\beta)^2}} \\ \rho_1 = \frac{rz_0/\sqrt{A}-\gamma}{\sqrt{(rx_0/\sqrt{A}-\alpha)^2+(ry_0/\sqrt{A}-\beta)^2}}, & \rho_2 = \frac{rz_0/\sqrt{A}+\gamma}{\sqrt{(rx_0/\sqrt{A}+\alpha)^2+(ry_0/\sqrt{A}+\beta)^2}} \end{cases}$$

3.3.3 Calcul de la longueur du cylindre

L'objectif de cette phase de la reconstruction est de déterminer la longueur du cylindre. La solution proposée ici pour obtenir cette longueur est de faire exécuter à la caméra un mouvement parallèle à l'axe D du cylindre. La distance parcourue par la caméra d'un bout à l'autre du cylindre sera égale à la longueur de celui-ci.

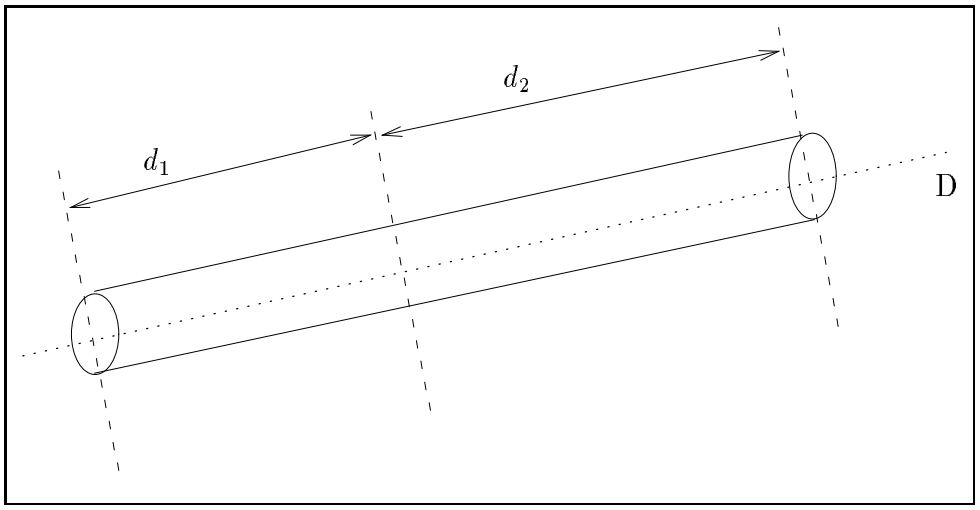


Figure 8 : calcul de la longueur du cylindre

Pour accomplir ce suivi, le robot doit exécuter une commande constituée de deux tâches :

- une **tâche principale** réalisée par asservissement visuel consistant à maintenir les deux limbes verticaux centrés dans l'image.
- une **tâche secondaire** consistant à décrire une trajectoire le long du cylindre suivant un axe D' parallèle à l'axe D du cylindre et tel que $d(D, D') = d$ avec $d > r$. Il s'agit en fait d'un mouvement de translation suivant l'axe $\vec{y}$ dans le repère de la caméra.

On s'arrêtera quand l'extrémité du segment sera au centre de l'image. Il faudra alors revenir à la position initiale du robot et repartir dans l'autre sens. Ainsi, en additionnant les deux distances parcourues par le robot, on disposera de la longueur du cylindre.

3.3.4 Reconstructions sommaires

L'objectif de ces reconstructions est de se donner des informations sur l'environnement de manière à optimiser les phases d'exploration et à faciliter par la suite le calcul des trajectoires en environnement incertain.

Soit $\mathcal{R}_d$ un processus de reconstruction qui permet de reconstruire une primitive de manière sommaire à partir d'un nombre limité d'images acquises par la caméra pendant la phase de reconstruction d'une autre primitive. La méthode de reconstruction utilisée sera celle présentée précedemment en utilisant la vision dynamique et non plus la vision active.

En effet, si on suppose la présence d'un segment (autre que celui sur lequel on est focalisé ou que ceux qui ont déjà été supprimés de la base de données 2-D) dans une séquence de n images, on dispose de la variation des caractéristiques $(\rho_i, \theta_i), i = 1 \cdots n$ de ce segment au cours du temps. Comme on connaît les positions ϕ de la caméra pendant cet intervalle de temps, on peut, grâce à l'algorithme de reconstruction 3-D du cylindre par vision dynamique, obtenir une première estimation des paramètres 3-D ($\underline{p}$) du cylindre.

Il faut noter que cette précision est toute relative car le mouvement de la caméra n'est pas commandé de manière à optimiser cette reconstruction. D'autre part, dans certaines configurations, cette estimation est impossible. En effet si on considère le cas de deux primitives perpendiculaires l'une par rapport à l'autre, le mouvement nécessaire à la reconstruction optimale de l'une ne pourra en aucun cas permettre une estimation même médiocre des paramètres de l'autre. En effet, le mouvement optimal de rotation autour de l'une des deux primitives se transforme pour la deuxième en un mouvement suivant la direction de son axe, ce qui ne permet pas l'estimation de ses paramètres. On ne reconstruira donc de manière sommaire que les primitives ayant une orientation proche de celle en cours de reconstruction, cela permettra d'eviter des résultats aberrants et donc inutiles sur les autres primitives. Signalons de plus que cette reconstruction est incomplète puisqu'elle ne permet pas le calcul de la longueur du cylindre.

Plusieurs processus de ce type peuvent être actifs en même temps. Ils sont déclenchés par l'occurence d'événements dans l'image : la reconstruction commence par la détection dans l'image d'un nouveau segment et se termine quand ce segment n'est plus présent dans l'image ou que la reconstruction de la primitive principale se termine.

3.4 Exploration de la scène

3.4.1 Objectifs

Une caméra, comme tous les systèmes de perception existant actuellement, n'est capable de percevoir qu'une infime partie des informations présente dans la scène étudiée. Outre le problème soulevé par l'absence possible dans le cône de vision d'un certain nombre d'objets de la scène, il en existe un autre dû aux occlusions dans l'image.

Pour résoudre ces problèmes, il est donc nécessaire d'observer la scène depuis plusieurs points de vue différents (chacun de ces points de vue donnera lieu à la création d'une base de donnée 2-D qui servira de base à la reconstruction des primitives présentes dans le cône de vision). Un système de vision doit donc être capable de déterminer automatiquement où et de quelle manière il doit se positionner de façon à obtenir de nouvelle information 3-D.

Il convient donc de développer des stratégies qui permettent d'obtenir la complétude par des prises de vue appropriées.

Zone observée Soit ϕ une position de la caméra et $S(\Omega)$ une carte du monde Ω à un instant donné, la zone de la scène observée depuis cette position est notée V_ϕ.
On définit l'ensemble $\mathcal{V}$ comme étant la zone de l'espace observée depuis le début de la reconstruction de la scène (voir figure 9c) tel que :

$$\mathcal{V} = \bigcup_{\phi} V_\phi(S(\Omega))$$

On peut aussi définir l'ensemble $\mathcal{V}$ après la n$^{\text{ième}}$ étape de reconstruction par la formule récurrente suivante :

$$\begin{cases} \mathcal{V}_{n+1} = \mathcal{V}_n \cup V_{\phi_n} \\ \mathcal{V}_0 = \emptyset \end{cases}$$

Dans le cas de la figure 9a où la caméra est dans la position ϕ (représentée par le point C) et observe une scène Ω composée de 7 cylindres verticaux, la partie V_ϕ de la scène observée est représentée en noir.

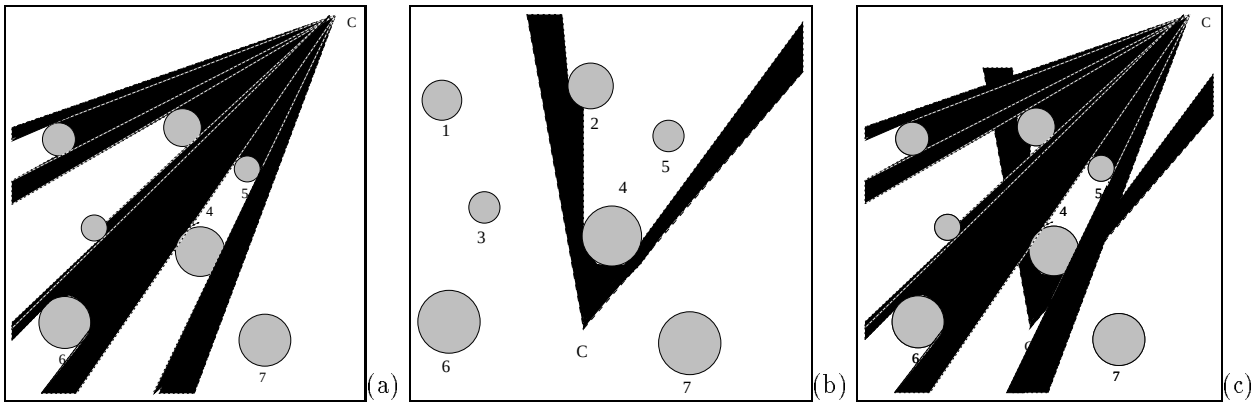


Figure 9 : Zones de visibilité V_{ϕ_1} et V_{ϕ_2} et $\mathcal{V} = V_{\phi_1} \cup V_{\phi_2}$

On constate ici, sur ces exemples, qu'un certain nombre de cylindres ne sont pas observés depuis la position ϕ de la caméra, soit parce qu'ils n'appartiennent pas au cône de visibilité de la caméra, soit parce qu'ils sont cachés par d'autres primitives.

Il est donc nécessaire de déplacer la caméra de manière à aller observer les zones n'appartenant pas à l'ensemble $\mathcal{V}$ de manière à assurer la complétude de l'exploration.

Exploration A partir d'une connaissance des zones de l'espace déjà explorées $\mathcal{V}$, d'une représentation aussi précise que possible de l'univers connu $S(\Omega)$, on définit une fonction exploration $\mathcal{E}$ qui va permettre à la caméra de se déplacer vers une nouvelle position définissant ainsi une nouvelle zone à reconstruire.

Le problème de l'exploration pose donc celui de la planification de trajectoires en univers incertain, celui de l'évitement d'obstacle ainsi que celui de la création de graphe de visibilité.

3.4.2 Calcul de la zone observée

Dans le but de déterminer la zone observée depuis une position ϕ de la caméra, on peut utiliser une des structures de donnée les plus fréquentes en géométrie algorithmique et en robotique : la triangulation de Delaunay.

3.4.2.1 Triangulation de Delaunay

Diagramme de Voronoï [10] [12] [24] Soit un ensemble $M = \{m_1, \ldots, m_n\}$ de n points de $\mathbb{R}^d$, le diagramme de Voronoï est un pavage de $\mathbb{R}^d$ constitué de cellules $C_1, \ldots, C_n$ définies par :

$$C_i = \{x \in \mathbb{R}^d : d(x, m_i) < d(x, m_j) \quad \forall j \neq i\}$$

où d désigne la distance euclidienne entre deux points de $\mathbb{R}^d$. Cette formule signifie que tout point x de la cellule C_i est plus proche du centre m_i de cette cellule que de n'importe quel autre point m_j.

Triangulation de Delaunay La triangulation de Delaunay est le dual géométrique du diagramme de Voronoi. On l'obtient en reliant par des arêtes les points $M = \{m_1, \ldots, m_n\}$ qui appartiennent à des cellules de Voronoï adjacentes. Parmi les propriétés de la triangulation de Delaunay, on peut citer les suivantes :

- c'est une triangulation qui maximise le plus petit angle des tétraèdres ;
- la sphère circonscrite à chaque tétraèdre ne contient aucun autre site ;
- elle minimise le volume total des sphères circonscrites.

3.4.2.2 Propriété de visibilité [17] [11] Une des propriétés intéressante de la triangulation de Delaunay est la propriété de visibilité : pour un segment de Delaunay AB, si O est le centre optique de la caméra, cette propriété indique que tout tétraèdre $QRST$ (où Q, R, S et T sont des points de la triangulation) intersecté par le triangle OAB peut être marqué comme étant visible depuis la caméra (figure 10).

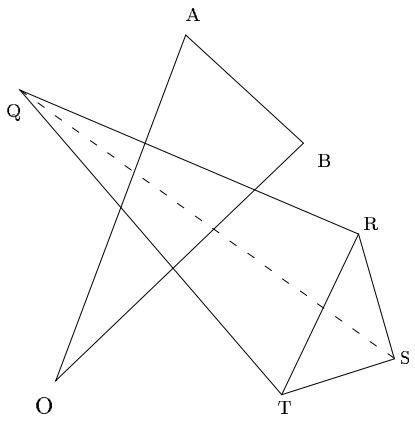


Figure 10 : Propriété de visibilité

En utilisant cette propriété, on peut construire facilement la zone V_ϕ de visibilité et en déduire aisément $\mathcal{V}$.

3.4.3 Objectif de complétude

L'un des problèmes que pose la reconstruction d'une scène Ω est qu'il ne faut pas oublier de reconstruire la moindre des primitives de la scène. Le nombre de primitives de la scène étant *a priori* inconnu, l'objectif de complétude de la reconstruction sera atteint lorsque il n'existera plus d'informations perceptibles par la caméra qui n'auront pas été traitées, c'est à dire si $\mathcal{V}_t = S$ (dans le cas où l'on étudie une scène située dans un espace borné $S \in I\!R^3$).

Remarque : Les idées présentées dans cette section n'ont pas encore été mises en œuvre. Ce problème concernant l'exploration est sans doute la difficulté majeure induite par la reconstruction globale de la scène. Il soulève en effet de multiples difficultés :

- la planification de trajectoire en milieu incertain ;
- l'exploration de la scène au moindre coût (au sens de la distance à parcourir par le robot).

3.5 Algorithme général de reconstruction

L'algorithme général suivant propose une méthode pour enchaîner les différentes phases de la reconstruction décrites précédemment.

Il montre l'alternance de phases d'exploration et de phases de reconstruction. A la fin de chaque phase d'exploration, on se construit une base de données 2-D, cette base contient une liste de segments de droite représentant les limbes des primitives de la scène. On accorde à chacun de ces segments un poids calculé en fonction de sa position dans l'image. Tous les segments de cette base de données vont être reconstruits de manière optimale dans l'ordre décroissant de leur poids. Parallèlement à ces reconstructions optimales, les primitives se projetant dans l'image avec une orientation proche de la primitive en cours de reconstruction seront elles-aussi reconstruites de manière sous optimale dans le but d'augmenter la connaissance de l'environnement (ces primitives sont celles correspondant aux autres segments contenus dans la base de données ainsi qu'à d'autres segments détectés ultérieurement par des observateurs extérieurs pendant la reconstruction). Cette connaissance du monde sera prise en compte pour mettre à jour le poids accordé à chaque segment de manière à minimiser la distance à parcourir pour la caméra. Quand tous les segments de la base de données initiale ont été traités (*i.e.* quand les primitives auxquelles ils étaient rattachés ont été reconstruites), on lance une nouvelle phase d'exploration devant permettre de trouver de nouvelles primitives. La reconstruction de la scène se terminera quand on sera sûr que toute la scène a été observée.

Initialisation de la zone visible à vide : $\mathcal{V} \leftarrow \emptyset$;
Initialisation de la carte du monde à vide : $S \leftarrow \emptyset$;
Tant que il existe une portion de l'espace non encore reconstruite **faire**
  Mise à jour de la zone déja observée : $\mathcal{V} \leftarrow \mathcal{V} \cup V_\phi$;
  Initialisation de la base de données $p(\omega)$;
  Calcul des poids de chaque segment de $p(\omega)$;
  Choix d'un segment dans $p(\omega)$;
  Pour tout segment de la base de données $p(\omega)$ **faire**
    (| Reconstruction optimale du cylindre
      | Reconstruire le cylindre avec un limbe;
      | Reconstruire le cylindre avec deux limbes ;
      | Calculer la longueur du cylindre ;
    | Reconstruction sommaire des autres primitives de $p(\omega)$
    ainsi que toute autre primitive se projetant dans l'image ;
    |)
    Mise à jour de la carte $S(\Omega)$;
    Mise à jour de la base de donnée $p(\omega)$;
    Réactualisation des poids des segments de $p(\omega)$;
    Choix d'un nouveau segment dans $p(\omega)$;
  fait
  Phase d'exploration - positionnement de la caméra : $\phi \leftarrow \mathcal{E}(\mathcal{V}, S)$;
  Calcul de la nouvelle zone observée depuis la position ϕ : $V_\phi \leftarrow V_\phi(S)$;
fait

Figure 11 : Algorithme de reconstruction d'une scène

Conclusion

L'algorithme de reconstruction proposé comporte une forte proportion de parallélisme. C'est pour cela que la mise en œuvre d'un tel algorithme par un langage impératif ne semble pas optimale. La définition d'un réseau d'automates, où plusieurs automates géreront les différentes tâches en parallèle, et sa programmation à l'aide du langage déclaratif synchrone SIGNAL vont maintenant être décrites.

4 Mise en œuvre par un réseau d'automates à l'aide du langage synchrone Signal

Dans le but d'enchaîner ces différentes tâches (reconstruction, exploration, ...), on souhaite mettre en œuvre un réseau d'automates permettant de gérer automatiquement la reconstruction globale de la scène.

4.1 Définition d'un réseau d'automates

Le réseau d'automates que l'on va définir devra être à même de gérer dans un premier temps trois tâches (voir figure 12) :

- **la reconstruction précise d'une primitive géométrique** ce qui inclut le calcul des paramètres ($\underline{p}$) de la primitive, et dans le cas particulier du cylindre, le calcul de la position du deuxième limbe ainsi que le calcul de la longueur du cylindre.

- **la reconstruction sommaire d'un certain nombre de primitives** ce qui correspond à l'acquisition des paramètres ($\underline{p}$) des autres primitives visibles par la caméra. Cette tâche doit s'exécuter en parallèle de la première.

- **l'exploration** qui consiste à amener la caméra vers une nouvelle position d'où l'on recommencera de nouvelles reconstructions.

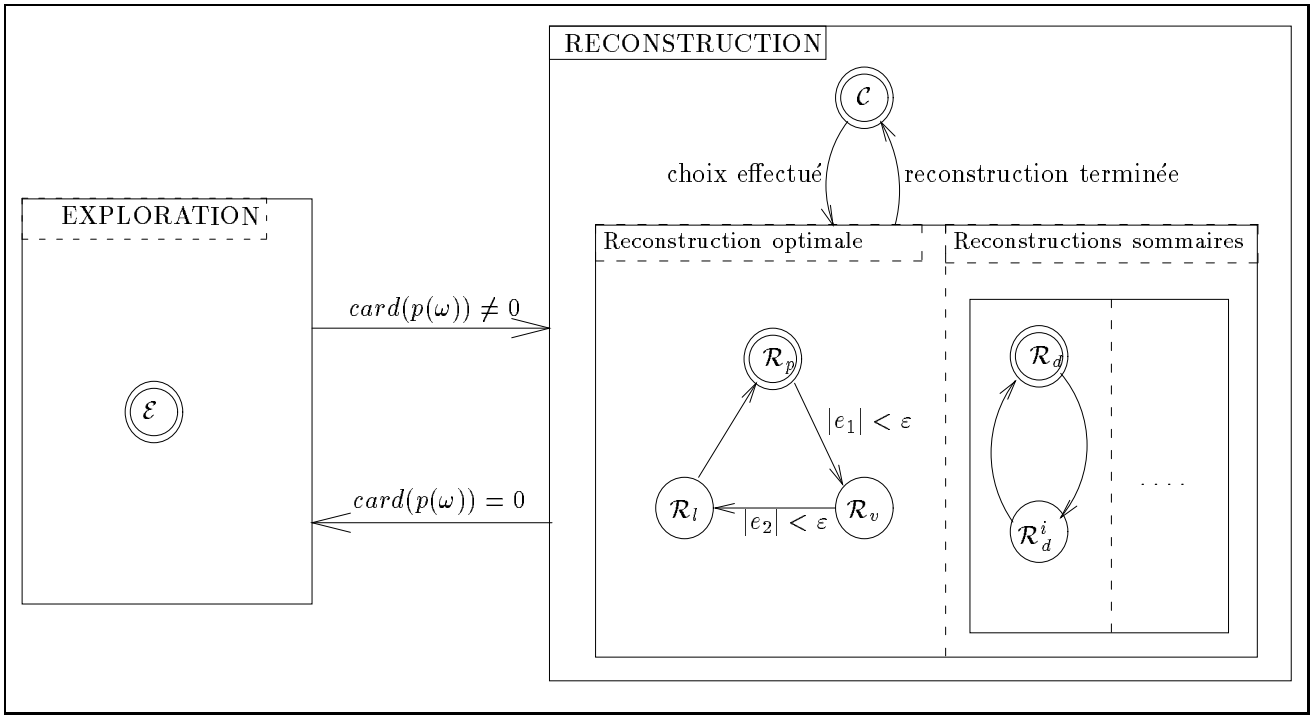


Figure 12 : Réseau d'automates

A chaque état d'un des automates, on associe une action à effectuer. Ce peut être aussi bien une action robotique (déplacement, calcul des paramètres par reconstruction en utilisant l'asservissement), qu'une action purement de calcul (reconstruction sommaire), ou que des actions de mise à jour des bases de données $S(\Omega)$ ou $p(\omega)$.

4.2 Le langage SIGNAL et les intervalles de temps

4.2.1 Une brève introduction à SIGNAL

Le langage SIGNAL, développé à l'IRISA [25] [26], est un langage synchrone de type flot de données et de style déclaratif. Il est construit autour d'un noyau restreint d'opérateurs de base. Ceux-ci manipulent des signaux, qui sont des suites non bornées de valeurs typées, dont une horloge associée détermine les instants auxquels la valeur est présente; par exemple, un signal `X` denote la séquence $(\mathtt{x}_t)_{t\in T}$ de données indexée par le temps t dans un domaine T. Des signaux d'un type particulier appelés `event` sont caractérisés seulement par leur horloge, c'est-à-dire leur présence (ils ont la valeur booléenne `true` à chaque occurrence). Étant donné un signal `X`, son horloge est donnée par l'expression `event X`, qui donne l'événement présent simultanément à `X`. Les constructeurs du langage permettent de spécifier dans un style équationnel des relations entre les signaux, c'est-à-dire entre leurs valeurs et entre leurs horloges. Des systèmes d'équations sont construits en utilisant la composition.

Le compilateur se livre à une analyse de la consistence du système d'équations, et détermine si les contraintes de synchronisation sont bien vérifiées. Si c'est le cas, et si le programme est contraint de façon à calculer une solution unique, alors un code exécutable est produit (en C ou en Fortran).

En SIGNAL, les opérateurs de base définissent des processus élémentaires, chacun correspondant à une équation :

Les opérateurs fonctionnels. Ils sont définis sur les types du langage (par exemple la négation booléenne du signal `E`: `not E`). Le signal (Y_t), défini par la fonction f dans: $Y_t = f(X_{1_t}, X_{2_t}, ..., X_{n_t})$ est écrit:

`Y := f{ X1, X2, ... , Xn}`

Les expressions fonctionnelles sont monochrones, ce qui signifie que les signaux `Y`, `X1`, ..., `Xn` sont dits synchrones : ils partagent la même horloge. En d'autres termes, pour calculer la valeur de Y_t, tous les X_i doivent être disponibles à l'instant t; pour cette raison ils sont contraints à avoir la même horloge, celle de Y.

Le retard. Il donne la valeur passée d'un signal, ce qui est généralement noté $ZX_t = X_{t-d}$, avec la valeur initiale $ZX_i = V_i$, for $0 \leq i < d$; en SIGNAL, pour le cas simple où $d = 1$, on écrit:

`ZX := X$1` avec l'initialisation `ZX init V0`

Le délai est monochrone lui aussi, c'est-à-dire que `X` et `ZX` ont la même horloge.

Le filtre. Le sous-échantillonnage d'un signal `X` selon une condition `C` est écrit :

`Y := X when C`

Cet opérateur est polychrone : les opérandes et le résultat n'ont pas la même horloge. Le signal `Y` est présent si et seulement si `X` et `C` sont présents au même instant et que `C` a la valeur `true`. Ainsi, `Y` est moins fréquent que `X` et que `C` à la fois : l'intersection des horloges de `X` et de `C` (c'est-à-dire les instants où l'expression peut être évaluée) inclut l'horloge de `Y` (qui ne comporte que les instants où `C` s'évalue à `true`). Quand `Y` est présent, sa valeur est celle de `X`.

La fusion. On définit la fusion de deux signaux `X` et `Y` par :

`Z := X default Y`

Cet opérateur est polychrone aussi : l'horloge de `Z` est l'union de celles de `X` et `Y`, elle est donc plus fréquente que chacune d'elles.

La valeur de `Z` est celle de `X` quand il est présent, sinon celle de `Y` quand il est présent.

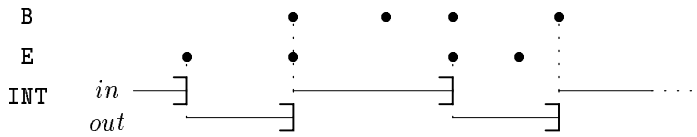


Figure 13 : Une trace possible d'un intervallle de temps avec ses bornes.

La composition. Les processus élémentaires peuvent être composés par l'opérateur commutatif et associatif "|" qui dénote l'union des systèmes d'équations. En SIGNAL, pour des processus P_1 et P_2 on écrit :

`(| P1 | P2 |)`

Par exemple, l'équation $x_t = x_{t-1} + 1$ peut s'écrire :

$$\begin{cases} x_t & = zx_t + 1 \\ zx_t & = x_{t-1} \end{cases}$$

ce qui s'écrit en SIGNAL : `(| X := ZX + 1 | ZX := X$1 |)`.

Par ailleurs, il est possible de confiner des signaux localement à un processus en utilisant "/" : par exemple dans l'exemple précédent, le confinement de `ZX` s'écrit :

```
(| X := ZX + 1 | ZX := X$1 |)/ZX
```

.

Le langage est construit autour de ce noyau et comporte des opérateurs dérivés pour les tableaux ou les variables par exemple. On peut définir des sortes de macro-instructions : des schémas de processus, qui ont un nom, des paramètres et des signaux d'entrée et de sortie typés, un corps et des déclarations locales. Les instances de schémas de processus rencontrées dans un programme sont expansées par un préprocesseur du compilateur.

Des opérateurs dérivés ont été définis à partir du noyau primitif pour alléger l'écriture de programmes en SIGNAL, tels que : `synchro{X,Y}` qui spécifie la simultanéité des signaux `X` et `Y` ; `when C` qui donne les occurrences de `C` à la valeur `true`; `X cell B` qui mémorise les valeurs de `X` et les produit aussi quand `B` est à `true`. On dispose aussi de tableaux de signaux et de processus.

4.2.2 Intervalles de temps et tâches

Comme on l'a vu plus haut, le langage SIGNAL a un modèle du temps fondé sur les instants, et les actions y sont instantanées. Or divers domaines d'application, tels que le traitement de signal ou le contrôle de processus physiques, requièrent l'enchaînement de différents mode de fonctionnement. Dans le cadre de cette étude en particulier, il s'agit de séquencer des tâches de vision suivant l'automate décrit en figure 12. À cette fin a été introduite la notion d'*intervalle de temps* [31], défini par des instants de début et de fin, et désignant la suite de ses occurrences. Nous donnons ici un bref aperçu de leur définition, et de la façon dont on peut les utiliser pour enchaîner des tâches.

4.2.2.1 Intervalles de temps. Un intervalle semi-ouvert à gauche[2] `INT`, aux événements de début `B` et de fin `E` est noté : `INT := ]B, E]`. Il dénote la suite de ses occurrences : il est à la valeur `inside` depuis l'occurence suivante de `B` jusqu'à l'occurence suivante de `E`; puis il est à la valeur `outside` jusqu'à l'occurrence suivante de `B`. On considérera par défaut, c'est-à-dire sauf mention contraire, que les intervalles sont initialement `outside`. La trace donnée en figure 13 illustre en particulier que, quand `B` et `E` sont simultanés, priorité est donnée au changement d'état.

Sur les intervalles on peut se définir des opérations comme l'union, l'intersection; le complémentaire `compl I` est défini comme l'intervalle dont la valeur est `outside` si `I` est `inside` et réciproquement. Le

[2]Les autres formes d'intervalles `[B,E[`, `[B,E]`, et `]B,E[` peuvent aussi être envisagées, mais nous choisissons `]B,E]` comme standard. En effet, dans un automate, la transition depuis un état de départ E_d vers un état d'arrivée E_a sur l'occurrence e d'un événement signifie que e arrive *dans* E_d, et qu'on n'est dans E_a qu'après la transition. En d'autres termes, causalement, la fin d'un intervalle se décide à l'intérieur de l'intervalle, alors que son début est externe.

filtrage des occurrences d'un signal `X` se trouvant à l'intérieur d'un intervalle `I` se note `X in I`; celles à l'extérieur sont obtenues par `X out I`. Par ailleurs, on peut aussi désigner leur bornes effectives, c'est-à-dire celles des occurrences des événements qui font changer l'état de l'intervalle: les événements d'entrée sont dénotés par `open I` défini par `B out I`, et les événements de fin `close I`, défini par `E in I`.

4.2.2.2 Tâches. Associer un intervalle de temps à un processus flot de données spécifie une *tâche*, c'est-à-dire une activité non-instantanée et son intervalle d'exécution. Quand elle entre dans une occurrence de son intervalle d'exécution, une tâche peut être :

- démarrée dans son état courant, tel que conservé quand elle a été suspendue : pour un processus `P` sur un intervalle `I` on note `P on I`.
- reprise à son état initial, si elle a été interrompue, ce qu'on écrit pour un processus `P` sur un intervalle `I` : `P each I`.

4.2.2.3 Enchaînements de tâches. Les intervalles permettent une référence explicite à des états et des transitions. Ils rendent possible la spécification d'automates ou de réseaux à places et transitions. A chaque état S_i correspond un intervalle de temps pendant lequel s'effectue la tâche associée. Enchaîner des tâches revient à mettre en séquence des intervalles sur l'occurence d'un événement. Dans le cas le plus simple, une transition d'un état `S1` à un état `S2` sur l'occurrence d'un événement `E`, comme illustré en figure 14, on peut écrire :

```
(|  S1 :=  ] A , E]
 |  S2 :=  ] E in S1, B] |)
```

ou encore :

```
(|  S1 :=  ] A , E]
 |  S2 :=  ] close S1, B] |)
```

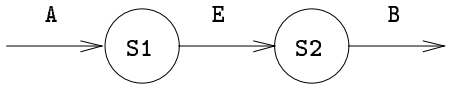


Figure 14 : Transition entre deux états

Ainsi, il est possible de spécifier des automates hiérarchiques, contrôlant le séquencement de tâches flot de données, qui peuvent elles-mêmes comporter un séquencement de sous-tâches.

4.2.3 Programmation des intervalles de temps en SIGNAL

L'intervalle de temps est un exemple de processus SIGNAL. On peut le coder par une variable d'état booléenne, où `true` code l'état `inside` et `false` code `outside`. La figure 15 illustre ce codage, obtenu avec l'interface graphique de l'environnement SIGNAL : les boîtes y correspondent à des niveaux de composition. La valeur de l'intervalle est donnée par `I`, et les événements donnant les occurrences des bornes sont donnés par `OPEN` et `CLOSE`.

4.3 Enchaînement de tâches

4.3.1 Intervalle d'éxécution des tâches

L'objectif de cette partie n'est pas de décrire ce que font les différentes tâches mais de décrire les intervalles de temps (en termes des évènements de début et de fin) pendant lesquels elles sont actives
.

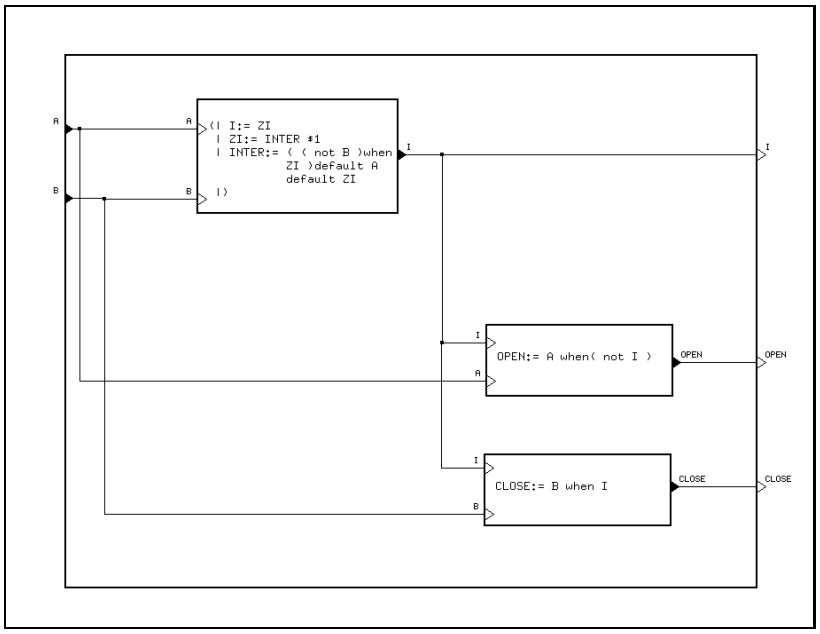


Figure 15 : Intervalle de temps programmé avec SIGNAL

Calcul des paramètres La reconstruction des paramètres d'une primitive commence s'il existe encore dans la base de données 2D des primitives à reconstruire (**when** $p(\omega) \neq \emptyset$) et si l'on n'est pas engagé dans une autre phase de reconstruction. Donc l'intervalle commence soit par le début de la reconstruction d'un ensemble de primitives (`open` $\mathtt{I}_{\mathtt{R}}$) soit par la fin de la reconstruction de la primitive précédente (`close` $\mathtt{I}_{\mathtt{R}_l}$). La reconstruction se termine quand on dispose des paramètres désirés avec une précision suffisante (`when` $|\mathtt{e}_1| < \varepsilon$).

Reconstruction basée sur les deux limbes Le but de cette partie de la reconstruction est de calculer la position du deuxième limbe, de vérifier si il est bien présent dans l'image puis de reconstruire le cylindre à partir des deux limbes. Soit $\mathtt{I}_{\mathtt{R}_v}$ l'intervalle de temps pendant lequel a lieu cette reconstruction :

On peut commencer cette phase de la reconstruction dès que la phase précédente est achevée (`close` $\mathtt{I}_{\mathtt{R}_p}$). Comme précédemment, la reconstruction se termine quand on dispose des paramètres désirés avec une précision suffisante (`when` $|\mathtt{e}_2| < \varepsilon$).

Calcul de la longueur L'objectif de cette phase de la reconstruction est de déterminer la longueur du cylindre. Soit $\mathtt{I}_{\mathtt{R}_l}$ l'intervalle de temps pendant lequel a lieu le suivi du cylindre. Cette phase peut commencer à partir du moment où la phase précédente est terminée. $f(p_\phi(\omega))$ est une fonction qui détecte l'occurence d'événements dans l'image. Elle se terminera quand on détectera l'extrémité du segment au centre de l'image.

```
Rec_optimale :  (|  (|   I_Rp := ] close R_l , precision_atteinte ]
                                    init inside
                    |   precision_atteinte := when |prec| < ε
                    |)
                |   (|   I_Rv := ] close I_Rp , segment_trouve ]
                    |   segment_trouve := when |prec| < ε'
                    |)
                |   (|   I_Rl := ] close I_Rv , longueur_calculee ]
                    |   longueur_calculee := when g(p(ω_φ))
                    |)
                |)
```

Reconstruction sommaire Soit $\mathcal{R}_s$ un processus de reconstruction qui permet de reconstruire une primitive de manière sommaire à partir d'un nombre limité d'images acquises par la caméra pendant la phase de reconstruction d'une autre primitive.

Plusieurs processus de ce type peuvent être actifs en même temps. Ils sont déclenchés par l'occurence d'événements dans l'image (présence d'un nouveau segment non repertorié dans l'image de base). Soit $\mathtt{I}_{\mathtt{R}_s}$ l'intervalle de temps pendant lequel a lieu cette reconstruction. La reconstruction commence par la détection dans l'image d'un nouveau segment (**seg_detecte**) et se termine quand ce segment n'est plus présent dans l'image (**seg_perdu**) ou que la reconstruction de la primitive principale est achevée (**close** $\mathtt{I}_{\mathtt{R}_l}$).

```
Rec_sommaire :  (|   I_Rs := ] (seg_detecte in I_R),
                                (seg_perdu in I_R) ]
                 |   Rec_somm each I_Rs
                 |)
```

Les reconstructions sommaires ont lieu en parallèle d'une reconstruction optimale d'une primitive :

```
Rec_primitive :  (| Rec_optimale
                  | Rec_sommaire_1
                  :
                  | Rec_sommaire_n
                  |)
```

La reconstruction d'une ensemble de primitive contenues dans une base de données 2-D $p(\omega)$ peut être vu comme l'alternance de phases de choix d'une primitive et de focalisation et, de phases de reconstruction de cette primitive.

```
Reconstruction :  (|   I_C := ] longueur_calcule,choisi ] init  inside
                   |   I_R := compl I_C
                   |   Choix each I_C
                   |   Rec_primitive each I_R
                   |)
```

Passage de l'automate de reconstruction à l'automate d'exploration On considère ici l'exploration comme se faisant en une seule étape. Ce processus resterait cependant à détailler.

L'exploration commence quand il n'y a plus d'éléments à reconstruire dans la base de donnée $p(\omega)$, elle se termine dès qu'une nouvelle position proposant de nouvelles primitives à reconstruire est trouvée.

```
(|   I_E := ] when p(ω) = ∅,
             when p(ω) ≠ ∅ ]
             init  inside
 |   I_REC := compl I_E
 |   Exploration each I_E
 |   Reconstruction each I_REC
 |)
```

4.3.2 Actions de mise à jour des bases de données

mise à jour de la base de données 2D La base de données 2D est mise à jour dans plusieurs circonstances :

- à la fin de la phase d'exploration quand la caméra se trouve devant une zone qui n'a pas encore été complètement observée. On peut associer cet instant à **close** $\mathtt{I}_{\mathtt{E}}$. Il s'agit ici d'initialiser la base de données.

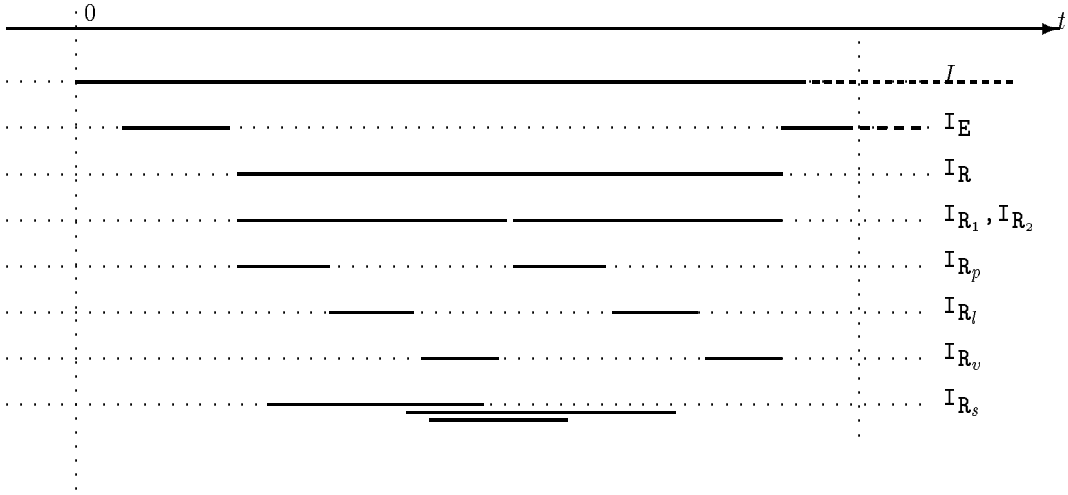


Figure 16 : Séquence possible d'intervalles de temps

- à la fin de la reconstruction du cylindre quand on connaît tous ses paramètres $\underline{p}$ ainsi que sa longueur. On peut associer cet instant à `close` I_{R_l}. Il s'agit là aussi d'une simple mise à jour : il faut supprimer les deux segments utilisés de la base de données 2-D.

mise à jour de la base de données 3-D La base de données 3-D est mise à jour à la fin de la reconstruction d'une primitive quand on connaît ses paramètres. On peut associer cet instant à `close` I_{R_l}. De même, elle est mise à jour à la fin de chaque phase de reconstruction sommaire (`close` I_{R_s}).

4.4 Intéret de l'utilisation de SIGNAL

L'hypothèse synchrone faite dans SIGNAL suppose que les actions que l'on désire exécuter ne prennent pas de temps (les entrées étant synchrones aux sorties). De ce fait, on suppose que la durée maximale d'une action est inférieure à la période de réaction de l'environnement, qui doit donner l'horloge de base du programme.

Or, dans l'état actuel des choses, cette hypothèse n'est pas applicable aux tâches considérées ici (reconstruction, exploration,...). Il faut donc sortir ces tâches du programme SIGNAL : on ne peut pas les considérer comme des opérateurs fonctionnels. Le programme final est donc divisé en deux parties : une partie synchrone (écrite en SIGNAL) gérant le réseau d'automates (voir figure 17) et donnant des ordres d'exécutions de certaines tâches robotiques, et une partie asynchrone exécutant les tâches demandées (écrite en C).

L'un des avantages d'un langage synchrone comme SIGNAL par rapport à d'autres langages est qu'il permet de faire des preuves de programmes et de vérifier certaines propriétés que l'on souhaite voir satisfaites par le programme. Ce langage permet en effet de spécifier facilement les propriétés que doit posséder le programme (cf §4.3) et de prouver qu'elles sont bien respectées à la mise en œuvre. La programmation du réseau d'automates en SIGNAL permet ainsi de montrer la compatibilité sur le plan temporel de l'agencement des différentes tâches.

D'autre part, la programmation en SIGNAL des différentes tâches robotiques elles-mêmes permettra de révéler les véritables avantages du langage pour ce type d'utilisation. Ces tâches sont pour l'instant exécutées par des processus externes asynchrones. Or leur comportement à flot de données, faisant des calculs sur des suites de données capteur, pour produire des suites de valeurs de commande, est dans le domaine d'application de SIGNAL en tant que langage de traitement du signal. En effet, à l'intérieur d'un programme SIGNAL on peut faire appel à des fonctions externes (écrites en C ou en FORTRAN) pour l'exécution de certains calculs (traitement d'images, calculs numériques ou matriciels) qui sont réalisés à l'intérieur de la boucle de commande. On peut donc considérer que ces fonctions ne

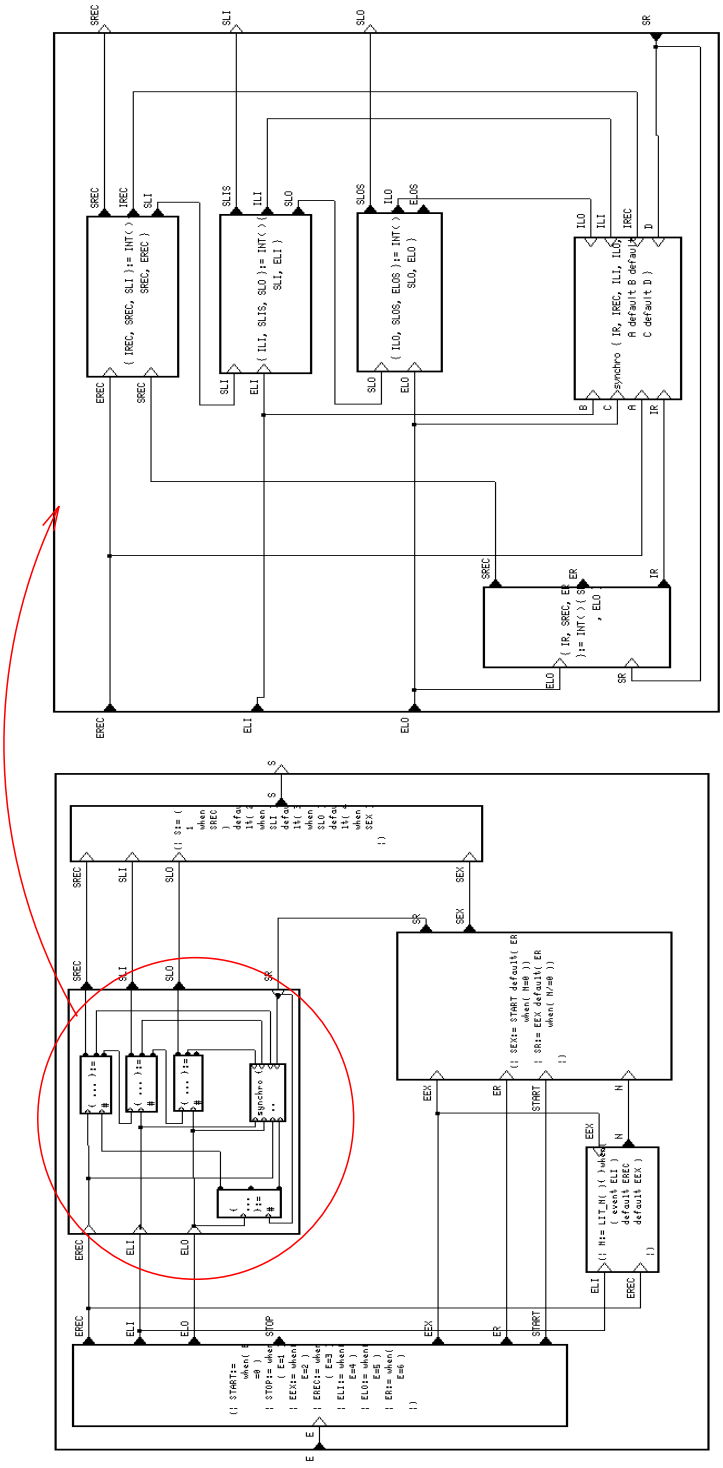


Figure 17 : Réseau d'automates programmé avec SIGNAL

prennent pas de temps par rapport au cycle de base. L'hypothèse synchrone peut alors être vérifiée et les possibilités offertes sont alors beaucoup plus importantes.

Le langage et son environnement fournissent alors toutes leurs possibilités pour la programmation des tâches : l'interface graphique, la description de l'algorithme indépendante de l'architecture, en particulier expression du parallélisme et sa gestion automatique par le compilateur, la modularité, la génération de code exécutable C, les outils de preuve de programme et de propriétés, ...

5 Résultats expérimentaux

Dans cette section, nous présentons les résultats qui ont été obtenus en simulation, la programmation sur un banc expérimental de vision robotique étant en cours de développement. Ces résultats sont basés sur l'utilisation d'un logiciel simulant le comportement d'une caméra embarquée sur un robot à 6 degrés de liberté.

5.1 Réseau d'automates

Le réseau d'automates présenté dans la section précédente est de fait constitué de deux automates : l'un gérant la reconstruction précise d'une primitive et l'autre devant gérer en parallèle la reconstruction sommaire des autres primitives non reconstruites présentes dans la base de données. Le processus de reconstruction décrit plus haut a ensuite été intégré à ce réseau, le calcul de la longueur du cylindre n'a pu être implémenté car le logiciel de simulation utilisé ne prend pas en compte la longueur des droites et des cylindres et les considère comme étant de longueur infinie.

En ce qui concerne l'automate de reconstruction sommaire, il est créé automatiquement à chaque nouvelle phase de reconstruction d'un ensemble de primitives. Les processus de reconstruction sommaire, eux, sont générés dynamiquement en fonction des besoins sur demande de l'automate de reconstruction principal ou sur l'occurence d'un événement extérieur.

L'automate d'exploration n'a pas été programmé pour le moment. On a pour l'instant considéré l'exploration comme une boîte noire et elle a pu de ce fait être intégrée sous la forme d'un état à l'automate de reconstruction.

Les communications entre l'automate et les tâches asynchrones se font par messages en utilisant les outils de communication inter-processus (IPC) mis à notre disposition par le système UNIX (sockets, mémoire commune).

5.2 Reconstruction optimale d'un ensemble de primitives

Une scène composée de trois cylindres (un cylindre horizontal, un autre vertical et le troisième incliné à 45 degrés, voir figure 18) a été reconstruite avec succès. La figure 19 montre l'évolution de

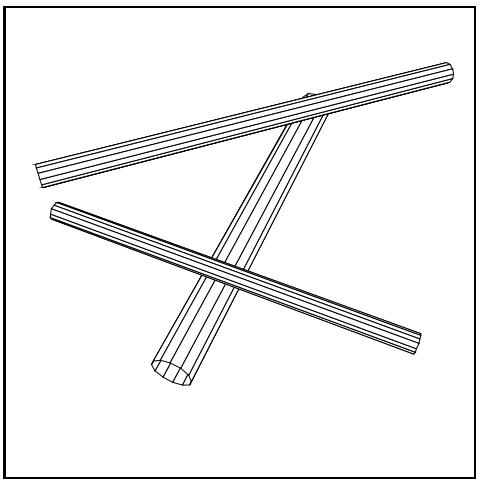


Figure 18 : *scène à reconstruire*

l'estimée du rayon, des paramètres de position des cylindres à reconstruire ainsi que les 3 composantes du vecteur directeur de l'axe des cylindres. On constate que chaque courbe est divisée en 3 parties correspondant à la reconstruction de chacun des 3 cylindres. Pour chaque cylindre et pour l'ensemble des paramètres, les valeurs, après une phase d'oscillation, convergent rapidement vers une position d'équilibre qui correspond à la valeur attendue du paramètre. Le processus de reconstruction a été déclenché pour chaque primitive depuis la même position initiale de la caméra correspondant à la position depuis laquelle on a créé la base de donnée $p(\omega)$.

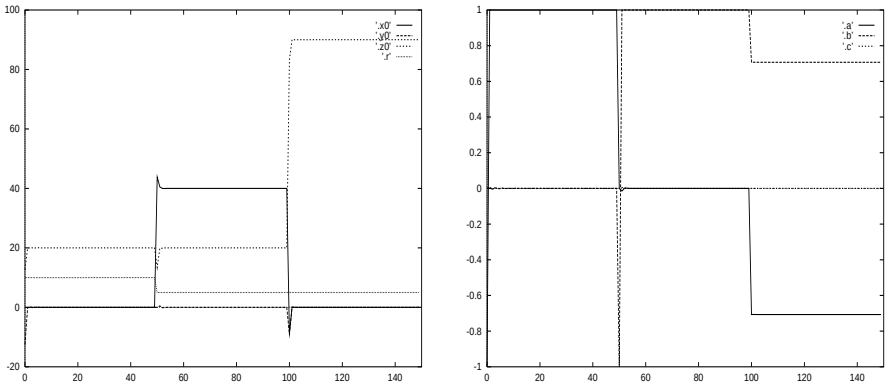


Figure 19 : *Reconstruction d'un ensemble de cylindres sans bruit. De l'itération 0 à 50 on reconstruit le premier cylindre, puis de 50 à 100 on reconstruit le deuxieme, enfin le troisième de l'itération 100 à 150.*

De manière à rendre cette simulation plus réaliste, un bruit artificiel à été introduit sur les données image ainsi que sur la vitesse de la caméra (figure 20).

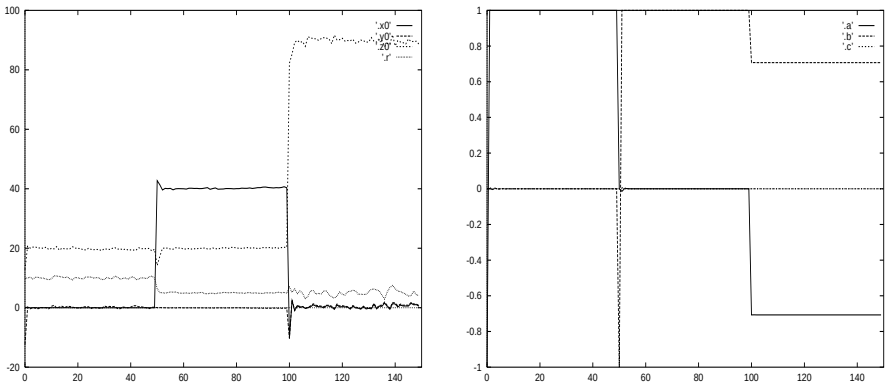


Figure 20 : *Reconstruction d'un ensemble de cylindres avec bruit*

Les transitions d'une primitive à l'autre se font correctement en respectant la méthode proposée dans la section 3.3. Il apparaît cependant que l'utilisation d'une matrice de transition à coefficients probabilistes, si elle rend un chemin de fixation proche de celui de l'œil, n'apporte pas grand chose de plus par rapport à l'utilisation d'une matrice déterministe. Par contre, il paraît important de pouvoir modifier cette matrice en fonction des informations au cours du temps.

5.3 Reconstructions sommaires

Parallèlement à ces reconstructions optimales des primitives, on a vu qu'il était utile de disposer d'une premiere estimée des paramètres des autres primitives de la scène. La scène qui a servi à tester ce module est composée de 3 cylindres dont deux sont verticaux et dont le troisième est inclinée de 22° par rapport à l'axe des deux premiers (Figure 21).

Les courbes de la figure 22 montrent l'évolution de l'estimation des paramètres de position des deux premiers cylindres quand la caméra est animée d'un mouvement de recontruction optimal succéssive-

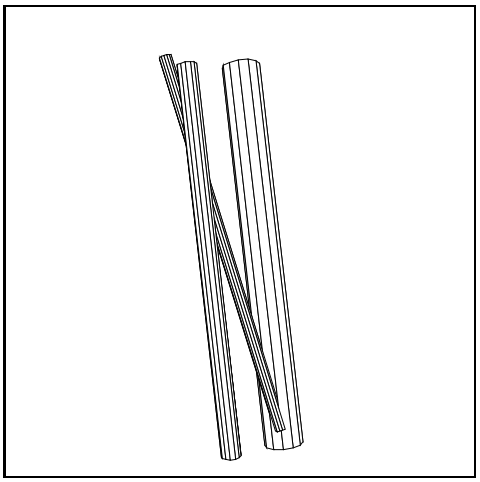
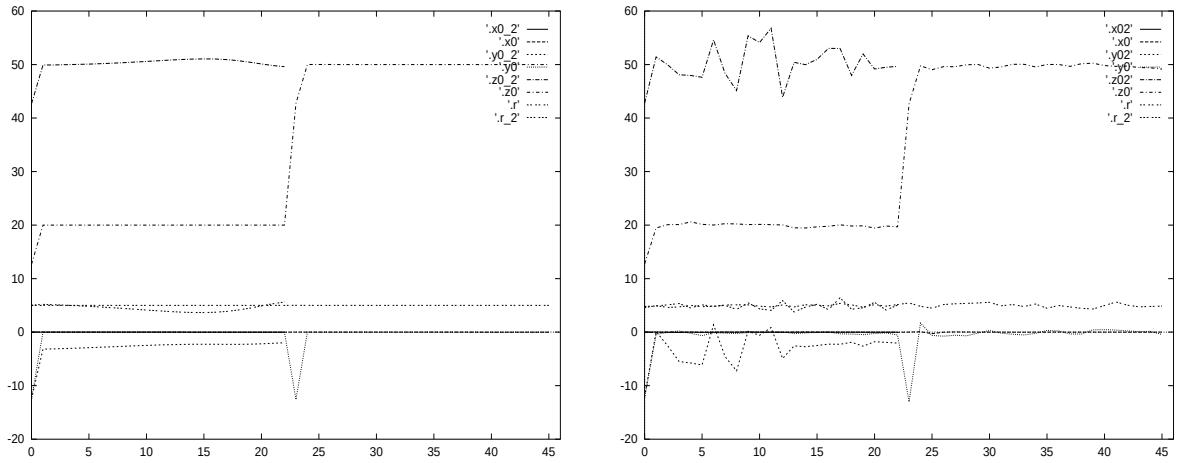


Figure 21 : *scène à reconstruire*

ment pour le premier puis le second cylindre. On a rajouté sur ces courbes l'évolution des paramètres de position du deuxième cylindre quand celui-ci est recontruit de manière sommaire (le mouvement de la caméra est optimal pour la recontruction du premier cylindre). On constate que l'estimée de ses paramètres (surtout si l'on introduit du bruit dans l'image et dans la position de la caméra) oscille autour des positions véritable de manière beaucoup plus importante que lors d'une reconstruction optimale.



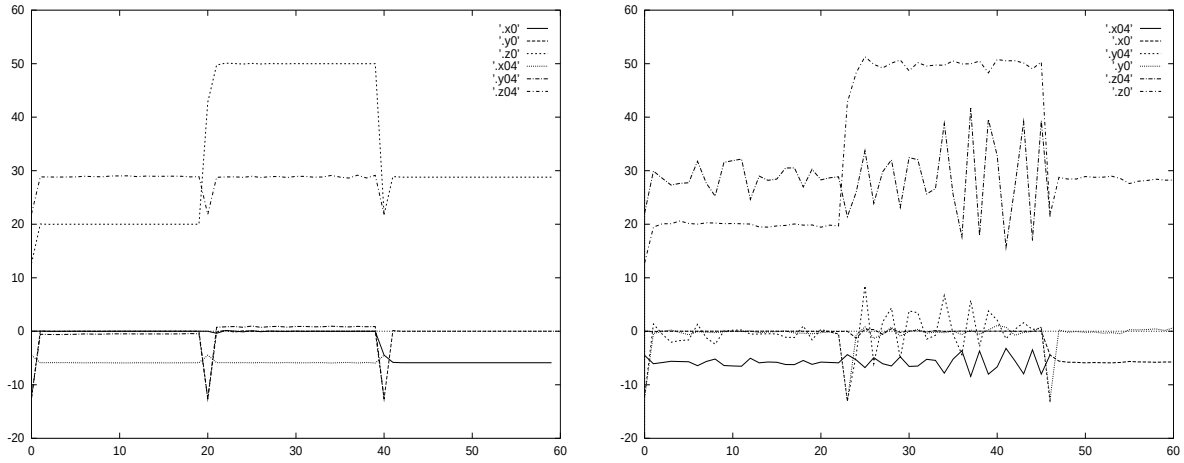
a) sans bruit b) avec bruit

Figure 22 : *Reconstruction optimale des deux premiers cylindres et reconstruction sommaire du deuxième. De l'itération 0 à 23 on reconstruit le premier cylindre de manière optimale par vision active) et le deuxieme de manière de manière sommaire par vision dynamique puis de l'itération 23 à 45, on recontruit le deuxième par vision active*

La figure 23 reprend en partie les résultats des courbes précédentes, mais pour le troisième cylindre. Comme prévu, la direction de l'axe de ce cylindre étant différente de la direction des deux autres, la reconstruction sommaire est nettement moins bonne, le problème étant surtout visible quand on introduit du bruit.

5.4 Reconstruction de droites

On a vu au § 3.3.1 que l'on pouvait aussi reconstruire des droites. Pour cela on a construit une scène (figure 24) constituée de deux cylindres verticaux et de trois droites formant un trièdre, la base des deux cylindres étant située sur l'un des plans qui composent le trièdre.



a) sans bruit b) avec bruit

Figure 23 : *Reconstruction sommaire du troisième cylindre. La reconstruction est ici optimale pour le premier cylindre de l'itération 0 à 23, puis pour le deuxième de 23 à 46, et enfin pour le troisieme de l'itération 46 à la fin. Parallèlement, le troisième cylindre est recontruit en utilisant la vision dynamique*

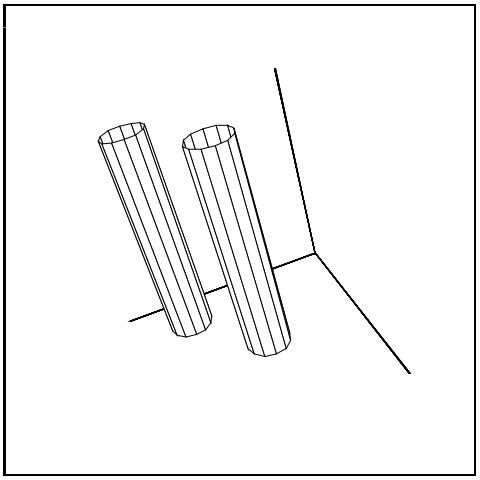


Figure 24 : *scène à reconstruire : droites et cylindres*

On a appliqué pour la reconstruction des droites l'algorithme de reconstruction des cylindres, le rayon estimé de ces cylindres étant nul, on en déduit que ce sont des droites. La courbe 25 montre l'évolution du rayon des primitives présentes dans la scène un bruit ayant été introduit artificiellement.

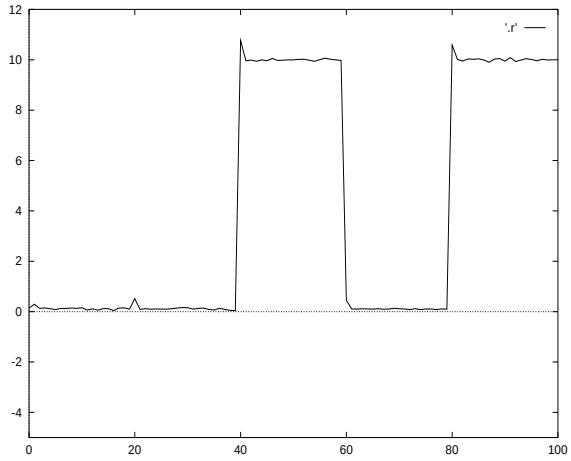


Figure 25 : *Reconstruction de droites et de cylindres avec bruit. De l'itération 0 à 20 puis 20 à 40, on reconstruit les deux premières droites on constate que l'estimation du rayon est presque nulle, on reconstruit ensuite un des deux cylindres de l'itération 40 à 60, puis, la dernière droite de l'itération 60 à 80 et enfin le second cylindre de l'itération 80 à 100.*

6 Conclusion

Dans ce rapport, nous avons étudié le problème de la reconstruction d'une scène tridimensionnelle composée de primitives paramétrables (ici, des cylindres et des droites).

Après avoir présenté une méthode de reconstruction de primitives 3-D basée sur la vision active et l'asservissement visuel, nous avons proposé, après avoir défini le problème de la reconstruction d'une scène complexe, une méthode permettant l'enchaînement des reconstructions axée sur les informations visuelles acquises dans l'image ainsi que sur les informations obtenues lors des reconstructions sommaires de la scène.

Cette méthode a été mise en œuvre par un réseau d'automates (écrit à l'aide du langage SIGNAL) qui gère automatiquement l'enchaînement des différentes phases liées à la reconstruction globale de la scène perçue, à savoir :

- la sélection des primitives à reconstruire. Un ensemble de segments extraits de l'image initiale constitue une base de données 2-D. Cette base de données permet de déterminer l'ordre de reconstruction, elle est mise à jour au fur et à mesure du processus de reconstruction 3-D ;
- la reconstruction précise une à une des primitives présentes dans la base de données 2-D ;
- la reconstruction sommaire (par vision dynamique) des autres primitives présentes dans le champ visuel de la caméra dans le but d'acquérir des informations complémentaires sur la scène.

Une ébauche du problème de l'exploration de la scène est donnée. La solution proposée, avec comme objectif d'assurer la reconstruction complète de la scène, repose sur un calcul de la zone de l'espace déjà observée par la caméra.

La reconstruction d'une scène complexe soulève encore un grand nombre de problèmes qu'il faudra résoudre :

- la possibilité de reconstruire d'autres types de primitives que les cylindres ou les droites : on pourrait entre autres envisager de reconstruire des sphères, des objets polyédriques, objets complexes avec des surfaces gauches et/ou des contours d'occultation...

- l'utilisation d'observateurs dans l'image : cela permettrait de détecter de nouvelles primitives dans l'image ;
- l'implémentation de la phase d'exploration : cette phase est sans aucun doute celle qui implique le plus de développement dans le futur, notamment sur les points suivants :
  - le calcul à l'aide de la triangulation de Delaunay et de la propriété de visibilité des zones de l'espace observées ;
  - la planification de trajectoire en univers incertain. Des méthodes axées sur la planification à l'aide de réseaux Bayésiens ou Markoviens peuvent entre autres être envisagées ;
  - la détermination d'une fonction d'exploration :
- la programmation en SIGNAL de l'ensemble du processus de reconstruction ;
- enfin, la mise en œuvre de ces processus sur une cellule expérimentale de vision robotique.

Remerciements

Ce travail a été réalisé dans le cadre d'une action commune du PRC Communication Homme Machine pole Vision (ORASIS) avec le projet ICARE de l'INRIA Sophia Antipolis.

Nous tenons aussi à remercier Samia Boukir, pour son aide.

References

[1] ADIV, G. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on PAMI*, Vol. 7, No. 4:pp. 384–401, Juillet 1985.

[2] ADIV G. Inherent ambiguities in recovering 3d motion and structure from a noisy flow field. *IEEE Trans. on PAMI*, Vol. 11, No. 5:pp. 477–489, Mai 1989.

[3] AGGARWAL J.K., WANG Y.F. Analysis of a sequence of images using point and line correspondences. *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1275–1280, Mars 1987.

[4] ALOIMONOS J. Purposive and qualitative active vision. *Proc. on* 10^{th} *ICPR, New Jersey*, Vol. 1:pp. 346–360, 1990.

[5] ALOIMONOS J., WEISS I., BANDOPADHAY .A. Active vision. 1^{st} *Int. Conf. on Computer Vision, London, England*, pp. 35–54, Juin 1987.

[6] ARBIB M.A., DIDDAY F.L. Eye movements and visual perception : a two visual system model. *INT. J. Man-Machine Studies*, 7:547-569, 1975.

[7] AYACHE N. *Vision stéréoscopique et perception multisensorielle. Collection science informatique*, InterEditions, Paris, France, 1989.

[8] BAJCSY, R. Active perception. *Proc. of the IEEE*, Vol 76, No. 8:pp. 996–1005, Aout 1988.

[9] BANDOPADHAY .A, CHANDRA B., BALLARD D.H. Egomotion using active vision. *Int. Conf. on CVPR 86*, pp. 498–503, Juin 1986.

[10] BOISSONNAT J. D. . Complexité géométrique et robotique. *Rapport de recherche INRIA*, Juin 1987.

[11] BOISSONNAT J.D., FAUGERAS O.D. LE BRAS-MEHLMAN E. Representing stereo data with the delaunay triangulation. *Artificial Intelligence Journal*, 44(1-2), juillet 1990.

[12] BOISSONNAT J.D., FAVERJON B., MERLET J.P. *Techniques de la robotiques : perception et planification.* Hermes, 1988.

[13] BOUKARRI B. Reconstruction 3-D récursive de scènes structurées au moyen d'une caméra mobile. Application à la robotique. *Thèse de l'université d'Orsay*, Octobre 1989.

[14] BOUKIR S. Reconstruction 3-D d'un environnement statique par vision active. *Thèse de l'université de Rennes I*, Octobre 1993.

[15] BOUKIR S., CHAUMETTE F., BOUTHEMY P., JUVIN, D. An active robot vision system for real-time 3-d structure recovery. *IFAC Workshop on Intelligent Autonomous Vehicles, Southampton, UK*, Avril 1993.

[16] BROWN C.M., RIMEY R.D. *Selective attention as sequential behaviour : Modeling eye movements with an augmented hidden markov model.* Technical Report, University of Rochester, Avril 1990.

[17] BUFFA M., FAUGERAS O., ZHANG Z. A stereovision-based navigation system for a mobile robot. *Rapport de recherche INRIA*, No 1895, Avril 1993.

[18] CHAUMETTE F. La relation vision-commande: théorie et application à des tâches robotiques. *Thèse de l'université de Rennes*, Juillet 1990.

[19] CHIEN C.H., AGGARWAL J.K. Model construction and shape recognition from occluding contours. *IEEE Trans. on PAMI*, Vol. 11, No. 4:pp. 372–389, Avril 1989.

[20] CROWLEY J.L, STELMASZYK P., PUGET P. Measurement and integration of 3-D structures by tracking edge lines. *Int. Journal of Computer Vision*, Vol. 8, No. 1:pp. 29–52, Juillet 1992.

[21] ESPIAU B., CHAUMETTE F., RIVES P. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, Vol 8, No. 3:pp. 313–326, Juin 1992.

[22] GARVEY T. Perceptual strategies for purposive vision. *Rapport de recherche SRI-AI Center*, N. 117, 1976.

[23] HACISALIHZADE S., STARK L. W. ,ALLEN J. S. Visual Perception and Sequences of Eye Movement Fixations: A Stochastic Modeling Approach. *IEEE Transaction On Systems, Man, and Cybernetics*, Vol 22(3):pp. 474–481, Mai-Juin 1992.

[24] KLEIN R. *Concrete and abstract Voronoi Diagrams. Lecture Notes in Computer Science, Vol 400*, Springer Verlag, 1989.

[25] LE GUERNIC P. ,GAUTIER T. ,BESNARD L. . SIGNAL : a declarative language for synchronous programming of real-time systems. In Gilles Kahn, editor, *Functional programming languages and computer architecture*, pages 257–277, Lecture Notes in Computer Science, 274, Springer-Verlag, 1987.

[26] LE GUERNIC P. ,LE BORGNE M. , GAUTIER T. , LE MAIRE C. Programming real time application with SIGNAL. In *Another look at real-time programming,*, special section of *Proceedings of the IEEE, 79(9)*, Septembre 1991.

[27] LUSTMAN F. Vision stéréoscopique et perception du mouvement en vision artificielle. *Thèse de l'Université d'Orsay*, Décembre 1987.

[28] NEGAHDARIPOUR S., HORN B.K.P. Direct passive navigation. *IEEE Trans. on PAMI*, Vol. 9, No. 1:pp. 168–176, Janvier 1987.

[29] NORTON D., STARK L. . Eye movements and visual perception. *Scientific American*, Vol. 224(6):pp. 34–43, Juin 1971.

[30] OKUTOMI M., KANADE T. A multiple-baseline stereo. *Proc. of CVPR 91, Lahaina, Maui, Hawaii*, pp. 63–69, 1991.

[31] RUTTEN E., LE GUERNIC P. The sequencing of data flow tasks in SIGNAL. *Rapport de recherche IRISA*, 778, Decembre 1993.

[32] SANDINI G., TISTARELLI M. Active tracking strategy for monocular depth inference over multiple frames. *IEEE Trans. on PAMI*, Vol. 12, No. 1:pp. 13–27, Janvier 1990.

[33] VERNON D., TISTARELLI M. Using camera motion to estimate range for robotic part manipulation. *IEEE Trans. on Robotics and Automation*, Vo. 6, No. 5:pp. 509–521, Octobre 1990.

[34] VIALA M. Contribution à la reconstruction de scènes constituées d'objets cylindriques et polyédriques à partir d'une séquence d'images acquises par une caméra en mouvement. *Thèse de l'université d'Orsay*, Novembre 1992.

[35] WAIN M.J. , STRICKER M.A. Promising direction in active vision. *Int. Journal of Computer Vision*, 11(2):pp. 109–127, Octobre 1993.

[36] WAXMAN A.M., PARSI B.K., SUBBARAO M. Closed-form solutions to image flow equations for 3D structure and motion. *Int. Journal of Computer Vision*, Vol. 1, No. 3:pp. 239–258, Octobre 1987.

[37] WELLS W.M. Visual estimation of 3D lines segments from motion. A mobile robot vision system. *IEEE Trans. on PAMI*, Vol. 5, No 6:pp. 820–825, Décembre 1989.

[38] WENG J., COHEN P., REBIHO N. Motion and structure estimation from stereo image sequences. *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 3:pp. 362–382, Juin 1992.

[39] WENG J., HUANG T.S., AHUJA N. Motion and structure from line correspondences : closed-form solution, uniqueness, and optimization. *IEEE Trans. on PAMI*, Vol. 14, N0. 3:pp. 318–336, Mars 1992.

[40] XIE M. Contribution à la vision dynamique: reconstruction d'objets 3D polyédriques par une caméra mobile. *Thèse de l'Université de Rennes 1*, Juin 1989.

[41] ZHANG Z. Analyse du mouvement à partir d'une séquence de scènes stéréoscopiques et applications à la robotique mobile. *Thèse de l'université d'Orsay*, Octobre 1990.

[42] ZHAO J., SHEN J., CASTAN S. Trinocular stereo vision by window matching in 3D space. *Proc. of 7^{th} SCIA Int. Conf. on Image Analysis*, pp. 202–209, 1992.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399